ORACLE9i

بالعربية

بسم الله الرحمن الرحيم

السلام عليكم ورحمة الله وبركاته:

هذا الكتاب للراغبين في القيام بإمتحانات الأوراكل 9 وبالتحديد للإمتحان الثاني رقم 1z0-031 **المسمى بإسم:**

Oracle 9i DBA Fundamentals I

عسى أن ينفعكم وبالتوفيق إن شاء الله تعالى.

أخوكم محمد.

المكونات:

الفصل الأول

مكونات الأوراكل

ORACLE ARCHITECTURE

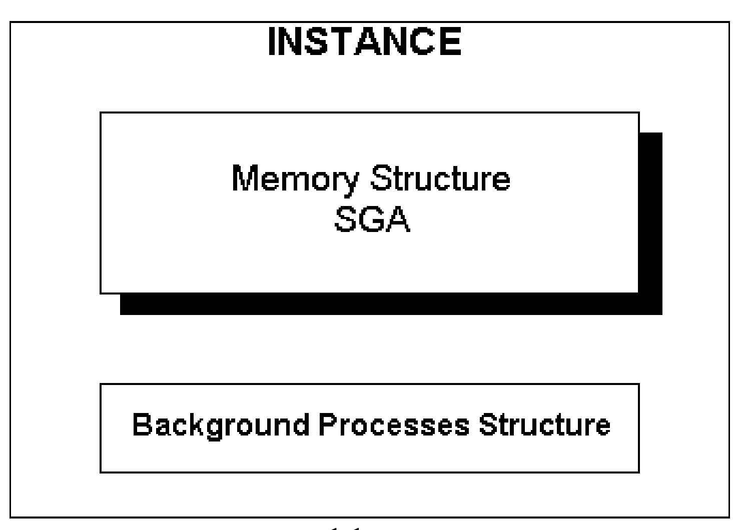
ORACLE SERVER

هو نظام لادارة قواعد البيانات و يتكون من مكونيين اساسيين هما :

ORACLE INSTANCE, ORACLE DATABASE

: ORACLE INSTANCE

من أهم مكونات الأوراكل يحتوي على Memory Structure و Process structure. لتتمكن من الوصول الى البيانات "Data" يجب أن يكون ال Instance في وضعية العمل، أي انه يتم من خلاله الحصول على البيانات المطلوبة، ولا يستطيع ال Instance فتح و تشغيل أكثر من Database واحد فقط في نفس الوقت، ولكن من الممكن لأكثر من Instance العمل على ذات ال Database.



رسم 1.1

: ORACLE DATABASE

عبارة عن مجموعة من الملفات لحفظ البيانات واستعادتها عند الطلب وينقسم الى قسمين هما Logical structure و Physical structure .

اما ال Logical يتكون من أقسام لتخزين وادارة ال Database .
اما ال Physical يتكون من الملفات الحقيقية الموجودة على الكمبيوتر أو السريفر .

مثال:

عندما تشاهد صورة على موقع انترنت فإنك تشاهد الجزء ال Logical أما الملف الحقيقي للصورة والذي مخزن في السيرفر يكون الجزء ال Physical. يكون هنالك ارتباط بين الجزئين بحيث اذا حذف أحدهما يجب حذف الجزء الأخر، اذ لا فائدة من إبقاء ملف الصورة ال Physical في السيرفر اذا كنت لا تريد عرض الصورة على الأنترنت.

ويتكون Database physical structure من ثلاث ملفات هي :

Control files: ملفات التحكم تحتوي على المعلومات اللازمة للمحافظة على ال Database ويجب ان يتوفر على الأقل ملف واحد ليعمل Oracle Database.

Data Files: الملفات التي يتم تخزين المعلومات والبيانات داخلها.

Redo Log Files: تقوم بتسجيل جميع المتغيرات التي طرأت على ال Database مثل أضافة أو حذف بعض البيانات "Data" لنتمكن من استعادة البيانات في حدوث ضياع بيانات بشكل مفاجئ (مثلاً: انقطاع التيار الكهربائي قبل حفظ البيانات المتغيرة).

ويوجد ملفات أخرى في ال physical ولكنها ليست ضمن ال Database انما ضمن ال Oracle Server.
ويجب أن تتواجد لتشغيل واستخدام واغلاق ال Database مثل:

Password File: الذي يحتوى على المعلومات اللازمة للدخول الى ال Instance مثل Privileges.

Archived Redo Log files: عبارة عن نسخ ل Redo Log Files للمزيد من الأمان في حالة حدوث أي ضياع للبيانات.

Parameter File: تخزن المواصفات الخاصة لل Instance مثلاً حجم الذاكرة الممنوحة لل Memory Structure ومواصفات أخرى نتطرق لها لاحقاً.

ORACLE SERVER PHYSICAL STRUCTURE

	ORACLE DATABASE
PARAMETER FILE	CONTROL FILES
PASSWORD FILE	REDO LOG FILES
ARCHIVED LOG FILES	DATA FILES

رسم 1.2

أما ال Logical Structure يتكون من:

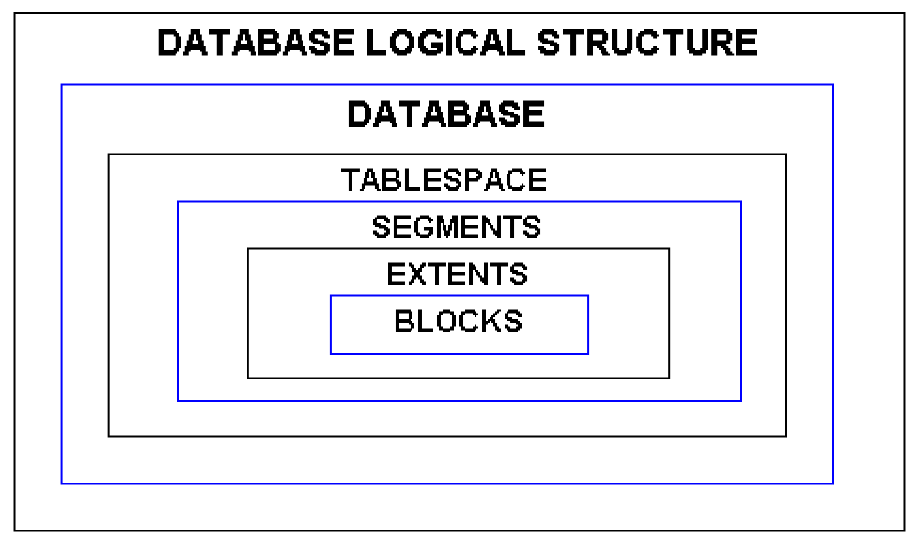
Tablespaces: تنقسم ال Database الى وحدات أصغر تسمى ال Tablespaces. كل Database يجب ان يتكون من واحد أو أكثر Tablespace. يوفر الأوراكل System Tablespace في بداية تكوين ال Database كحد أدني ضروري لعمل ال Database.

Segments: ينقسم ال Tablespace الى وحدات أصغر تسمى ال Segments.

Extent: ينقسم ال Segments الى وحدات أصغر تسمى ال Extents.

Blocks: تنقسم ال Extent الى وحدات أصغر تسمى ال Blocks وهي أصغر وحدات تخزين و قراءة البيانات.

سوف يتم التطرق لاحقاً بالتفاصيل في الكتاب لمكونات ال Logical Structure.



رسم 1.3

ORACLE MEMORY STRUCTURE

يتكون من منطقتين تعرفان باسم :

System Global Area (SGA): وهي ال Memory structure الخاص بال Instance الذي تم التحدث عنه مسبقاً (راجع الرسم 1.1) والتي هي أحد أهم مكونات ال Instance وتحجز المساحة الخاصة لها من الذاكرة (أو تبدأ بالعمل) عند تشغيل ال Instance. وهي ذاكرة مشتركة بين الأوامر "Processes" التي تأتي لل Instance مثل SQL Query Process وايضاً مشتركة بين مستخدمي ال Database المختلفين، وتعرف ايضاً باسم : *Shared Global Area*.

Program Global Area (PGA): وهي المساحة المخصصة من الذاكرة ل User Process وتحتوي على معلومات حول ال Server Process. وهي ذاكرة خاصة لأمر "Process" واحد فقط. وتعرف ايضاً باسم : *Private Global Area* أو *Process Global Area*.

سوف يتم التطرق لاحقاً ل User Process و Server Process.

Oracle Memory Strcuture

SGA

ذاكرة مشتركة

PGA

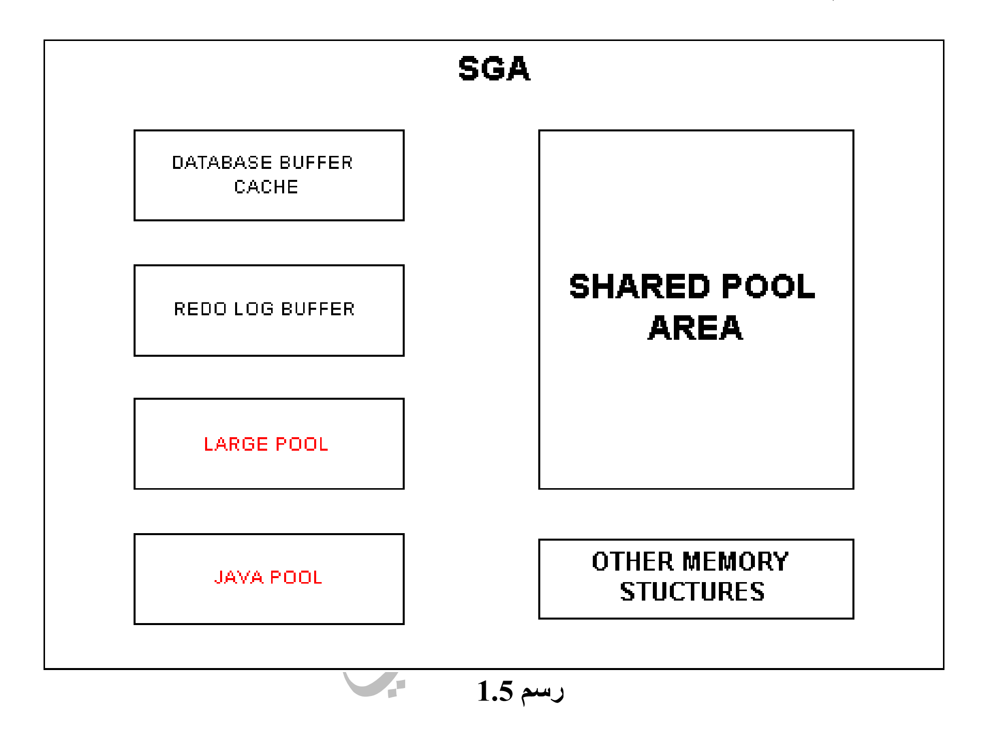
ذاكرة خاصة

رسم 1.4

SYSTEM GLOBAL AREA (SGA)

جميع مستخدمي ال Database يتشاركون البيانات الموجودة في هذه المنطقة حيث يتم تخزين البيانات المشتركة من مختلف الأوامر "Database Processes" لكي تسهل عملية استخراج البيانات . يقوم ال Oracle بحجز المساحة المخصصة له من الذاكرة عند بداية تشغيل ال Instance ويقوم بتحرير المساحة عند انهاء عمل ال Instance، وتقسم الى عدة أقسام منها أقسام اساسية ضرورية ومنها أقسام اختيارية.

الرسم رقم 1.5 يوضح الأقسام الاساسية والأقسام الاختيارية (الأقسام الاختيارية باللون الاحمر)، وسوف يتم التطرق الى كل قسم على حده.



رسم 1.5

تعتبر ال SGA ذاكرة مرنة ديناميكية أي ان باستطاعة أقسامها ان تكبر أو تصغر في الحجم دون أغلاق ال Instance لأسباب مختلفة مثل كثرة العمل (كثرة الأوامر) على أحد أقسامها اذ أن زيادة العمل على احد الأقسام يتتطلب ذاكرة اضافية ولكن لا يمكن للذاكرة العامة لل SGA ان تتخطى الحد الأعلى المحدد بالعامل "Parameter" SGA_MAX_SIZE.

ملاحظة:SGA_MAX_SIZE هي احد مكونات ال Initialization Parameter File التي سوف يتم التطرق لها لاحقاً.

تذكر: ال Parameter File التي جاء ذكرها قبل قليل.

مثال تطبيقي 1.1:

لمعرفة حجم ال SGA الحالي اكتب التالي في برنامج SQLPLUS:

```
SHOW SGA;
```

ال *Total System Global Area* تمثل حجم ال SGA الحالي.

مثال:

اذا كانت الذاكرة المخصصة لل SGA حوالي 100 ميغا بايت والتي تعتبر "SGA_MAX_SIZE" وكان التوزيع المبدئي للذاكرة على الأقسام المختلفة على النحو التالي:

Shared Pool Area= 50 ميغا بايت
Database Buffer Cache= 25 ميغا بايت
Redo Log Buffer= 10 ميغا بايت (منطقة ثابتة غير متغيرة)
باقي الأقسام = 15 ميغا بايت. (الأقسام المتغيرة)
الذاكرة الكاملة = 100 ميغا بايت

وزاد ضغط العمل على Shared Pool Area بحيث أن 50 ميغا بايت لم تعد تكفي، فبمقدور Shared Pool Area بان تأخذ ذاكرة اضافية من باقي الأقسام ولكن لا يمكن للذاكرة الكاملة بأن تزيد عن 100 ميغا بايت.

Shared Pool Area= 60 ميغا بايت
Database Buffer Cache= 20 ميغا بايت
Redo Log Buffer=10 ميغا بايت (منطقة ثابتة غير متغيرة)
باقي الأقسام = 10 ميغا بايت (الأقسام المتغيرة)
الذاكرة الكاملة = 100 ميغا بايت

الذاكرة في ال SGA عبارة عن وحدات متواصلة تسمى Granules وحجم ال Granule يعتمد على SGA_MAX_SIZE.

SGA_MAX_SIZE أصغر من 128 ميغا بايت, اذاً حجم كل Granule يساوي 4 ميغا بايت أما اذا كان حجم ال SGA_MAX_SIZE اكبر من 128 ميغا بايت اذاً حجم كل Granule يساوي 16 ميغا بايت.

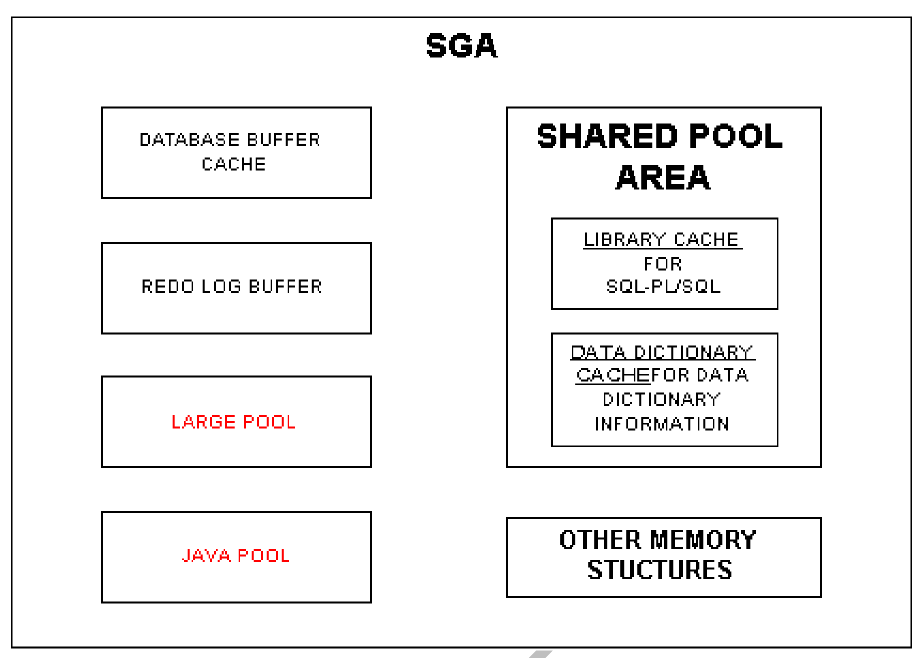
الحد الأدني من عدد ال Granules لل SGA هو ثلاث.

واحد Granule للمنطقة الثابتة في SGA مثل Redo Log Buffer
واحد Granule ل Shared Pool Area
واحد Granule ل Database Buffer Cache

ملاحظة: يمكن الحصول عن بيانات عن ال Granules من V$BUFFER_POOL.

:SHARED POOL AREA

تستخدم لحفظ أخر أو أحدث أوامر ال SQL و PL/SQL وأخر البيانات(المعلومات) المستخرجة من ال Data Dictionary. تتكون من قسمين هما : *Library Cache* و *Data Dictionary Cache*.



رسم 1.6

باعتبارها منطقة مهمه جداً فمن الممكن تغير حجمها ديناميكياً (بدون اغلاق ال Instance) بحيث لا يتجاوزالزيادة في المساحة مساحة ال SGA المحدده بالعامل "Parameter" SGA_MAX_SIZE.
يعتبر حجم ال SHARED POOL AREA محددة من قبل العامل "Parameter" SHARED_POOL_SIZE

مثال تطبيقي1.2:

لمعرفة الحجم الحالي ل Shared Pool Area اكتب التالي:

```
SHOW PARAMETER SHARED_POOL_SIZE;
```

في الكمبيوتر الخاص بي الحجم هو : 46 MB و يظهر كالتالي = 46137344

لتغيير مساحة ال Shared Pool Area ديناميكياً اكتب التالي:

```
ALTER SYSTEM SET SHARED_POOL_SIZE= 64M;
```

في حالة نجاح الأمر يظهر الجواب التالي من SQL PLUS : *System altered*.

أما في حالة عدم وجود ذاكرة أضافية لان تضاف ل Shared Pool Area يظهر الجواب التالي:*Insufficient memory to grow*.

:Library Cache

تحتوي على :

- المنطقة رقم 1: SQL Shared Statements
- المنطقة رقم 2: PL/SQL Procedures or Packages

مثال:

عندما يقوم مستخدم لل Database بطلب جملة ال SQL فإن الجملة وطريقة انجاز مهامها تخزن في المنطقة رقم 1 وبذلك يسهل عملية تكرار انجاز "Execute" ال SQL من الذاكرة بسرعة أكبر في حال تم طلب نفس الجملة من مستخدم أخر وبذلك تكون العملية أسرع وتخفف العبئ على ما يعرف باسم Compilations، وكذلك ينطبق الحال على PL/SQL.

خطوات عملية:

الخطوة 1: يقوم المستخدم الأول بطلب جملة ال SQL التالية : select * from employees;
الخطوة 2 : يقوم ال Server Process بدراسة الجملة ومعرفة المراحل التي سوف يتم بعدها انجاز"Execute" الجملة على اعتبار أن هذه الجملة ليست مخزنة في ال Library Cache.
الخطوة 3: يتم تخزين الجملة ومراحل انجازها في Library Cache.
الخطوة 4: يتم اظهار البيانات الناتجة من جملةال SQL للمستخدم الأول، وبذلك تكون جملة ال SQL تم انجازها وتم تسجيل جميع المراحل التي مرت بها الجملة في ال Library Cache.
الخطوة 5: يقوم المستخدم الثاني بعد قليل بكتابة نفس جملة ال SQL وهي select * from employees;
الخطوة 6: يقوم ال Server Process بدراسة الجملة فيجد انها موجوده في ال Library Cache فينفذ مراحل انجاز الجملة دون دراسة الجملة باعتبار ان المراحل التي تمت لانجاز الجملة مخزنة في ذاكرة ال Library Cache.
الخطوة 7: يتم اظهار البيانات المطلوبة للمستخدم الثاني بسرعة أكبر.

:Data Dictionary Cache

المكون الثاني ل SHARED POOL AREA والتي يحتوي على بيانات من ال Data Dictionary حول: ...Tables, Indexes, Privileges, etc. وهي تعمل بنفس طريقة ال Library Cache .

وعندما تمتلئ منطقة Library Cache أو Data Dictionary Cache يقوم الأوراكل باخراج أقدم بيانات لم يتم تكرار طلبها لتعوض ببيانات جديدة وتعرف الطريقة باسم: Least Recently Used (LRU)

:DATABASE BUFFER CACHE

يتم تخزين فيها أحدث البيانات التى تم استخراجها من الملفات الفيزيائية "Data Files" ، وفي حالة طلب ذات البيانات من ذات المستخدم أو من مستخدمين أخريين لل Database يتم استخراج البيانات من الذاكرة وليس من Data Files.

ملاحظة: تتم ادارتها ايضاً بنظام Least Recently Used (LRU).

مثال:

عندما يقوم مستخدم لل Database بطلب بيانات محددة من ال Database عن طريق مثلاً جملة ال SQL فإن البيانات المستخرجة من ال Data Files تخزن في ال Database Buffer Cache وبذلك يسهل عملية استخراج البيانات وبسرعة أكبر في حال تم طلب نفس الجملة من ذات المستخدم أو مستخدم أخر وبذلك يخف العبئ على ما يعرف باسم Input/Output Load.

راجع الرسم 1.7 لمزيد من التوضيح.

الخطوات المبينة على الرسم 1.7:

الخطوة 1: يقوم المستخدم الأول بطلب بيانات من الDatabase .
الخطوة 2: يقوم ال Server process بدراسة الطلب ويحضر ال Server Process البيانات من الData Files على اعتبار أن هذه البيانات المطلوبة ليست مخزنة في ال Database Buffer Cache.
الخطوة 3: يتم احضار البيانات من ال Data Files وتخزن البيانات في Database Buffer Cache.
الخطوة 4: يتم اظهار البيانات المطلوبة للمستخدم الأول.
الخطوة 5: يقوم مستخدم أخر بعد قليل بطلب ذات البيانات من ال Database.
الخطوة 6: يقوم ال Instance بدراسة الطلب فيجد ان البيانات المطلوبة تم استخراجها قبل قليل من ال Data Files ومازالت مخزنة في ال Database Buffer Cache فيستخرج البيانات من ذاكرة ال Database Buffer Cache دون الحاجة الى استخراج البيانات من القرص الطلب حيث توجد ملفات Data Files، ومن ثم يتم اظهار البيانات للمستخدم الثاي بسرعة أكبر وجهد أقل.

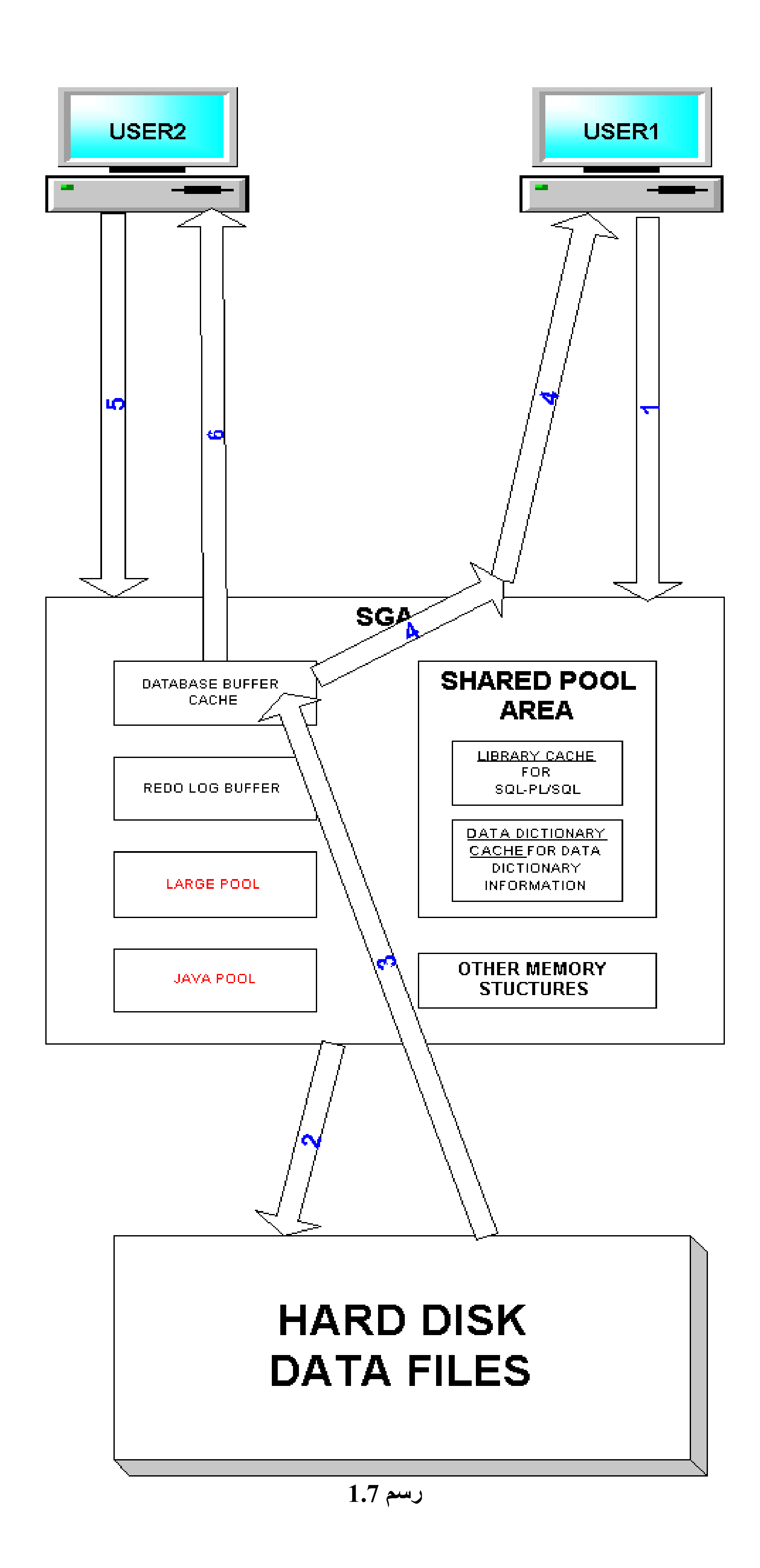
وتتكون ال Database Buffer Cache من ثلاثة أجزاء مستقلة تسمى Sub Cache:

Default: يخزن في هذا القسم البيانات التي ليست من ضمن الأقسام الأخرى (Keep, Recycle). ويتحكم بمساحتها العامل "Parameter" DB_CACHE_SIZE.

Keep: تحافظ على البيانات المخزنة ولا تستبدل، ويتحكم بمساحتها العامل "Parameter" DB_KEEP_CACHE_SIZE.

Recycle: يتم ازالة البيانات من هذه المنطقة عند عدم الحاجة اليها مجدداً، ويتحكم بمساحتها العامل "Parameter" DB_RECYCLE_CACHE_SIZE.

ملاحظة: مساحة ال Default لا يمكن أن تكون ان تساوي صفر أبداً.


USER2
USER1
5
6
4
1
SGA
4
DATABASE BUFFER CACHE
SHARED POOL AREA
REDO LOG BUFFER
LIBRARY CACHE FOR SQL-PL/SQL
DATA DICTIONARY CACHE FOR DATA DICTIONARY INFORMATION
LARGE POOL
JAVA POOL
3
OTHER MEMORY STUCTURES
2
HARD DISK DATA FILES

رسم 1.7

مثال تطبيقي 1.3:

لمعرفة أحجام أقسام ال Database Buffer Cache اكتب التالي:

SHOW PARAMETER DB_CACHE_SIZE;

لتغيير مساحة ال Database Buffer Cache ديناميكياً اكتب التالي:

ALTER SYSTEM SET DB_CACHE_SIZE= 30 M;

Data Block and Data Buffer:

البيانات المخزنة في ملفات الأوراكل تكون مخزنة بشكل كتل ولذلك يطلق عليها اسم **Blocks** أما البيانات المخزنة في ال Database Buffer Cache فتعرف باسم **Data Buffer**. وحجم ال Buffer هو نفسه حجم Block والمحدد بالعامل "Parameter" **DB_BLOCK_SIZE**.

Data Buffer Advisory Parameter (DB_CACHE_ADVICE):

يساعد مدير البيانات (DBA) على ادارة ال Database Buffer Cache وخاصة في أوقات الذروة حيث يقوم باظهار احصائيات ومعلومات عن مختلف ال Database buffer cache .
يمكن الحصول على البيانات التي ينتجها العامل "Parameter" من **V$DB_CACHE_ADVICE**.
وله ثلاث حالات هي:

OFF: لا يوجد مساحة في الذاكرة مخصصة له وبالتالي لا يقوم بتكوين أي بيانات
ON: يقوم بتكوين البيانات وله مساحة مخصصة في الذاكرة.
READY: يوجد مساحة في الذاكرة مخصصة له ولكنه في وضعية الاستعداد ولا يقوم بتكوين اي بيانات.

ربما تتسائل لماذا اذاً توجد الحالة **READY** لماذا لم يتم الاكتفاء بأول حالتيين فقط؟
الجواب على ذلك يكمن في ان انتقال العامل "Parameter" من وضعية ال OFF الى ON قد يؤدي الى أخطاء في الذاكرة أو فشل عملية التحويل.

مثال تطبيقي 1.4:

لمعرفة الوضع الحالي اكتب التالي:

SHOW PARAMETER DB_CACHE_ADVICE;

يمكن تغيير وضيعة العامل "Parameter" **DB_CACHE_ADVICE** ديناميكياً بكتابة :

ALTER SYSTEM SET DB_CACHE_ADVICE = READY;

ملاحظة: قد تفشل المحاولة اذا كان الوضع الابتدائي **OFF**.

:REDO LOG BUFFER

من أهم اسس حماية البيانات اذ تحتوي على بيانات عن جميع المتغيرات التي طرأت على ال Database مثل التغيرات الناتجه عن أوامر INSERT,DELETE,ALTER,DROP, CREATE.
المتغيرات التي حدثت في ال Database تسجل في ال Redo Log Buffer باسم Redo Entries.
مساحة ال Redo Log Buffer محددة من قبل العامل "Parameter" LOG_BUFFER.

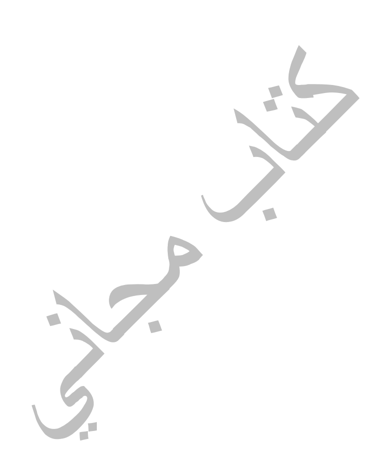
مثال تطبيقي 1.5:

لمعرفة الحجم الحالي اكتب التالي:

```
SHOW PARAMETER LOG_BUFFER;
```

لا يمكن تغيير وضيعة العامل "Parameter" LOG_BUFFER ديناميكيا.

تذكر: ان Redo Log Buffer منطقة ثابتة غير متغيرة.



:LARGE POOL

هي منطقة ذاكرة اختيارية وليست اساسية (اجبارية) يمكن لل DBA ان يقوم بتشكيلها حين الحاجة اليها حيث توفر ذاكرة كبيرة لعمليات عديدة مثل BACKUP and RESTORE.

ملاحظات:

1- علمية ال BACKUP هي العملية التي يقوم فيها ال DBA بحفظ نسخ "Copies" من البيانات غالباً على أقراص صلبة Hard disk لكي يتمكن من المحافظة على البيانات واسترجاعها RESTORE عند الحاجة.

2- ال Large Pool لا تعمل بنظام ال LRU.

3- مساحة ال Large Pool محددة من قبل العامل "Parameter" LARGE_POOL_SIZE.

4- يمكن تغير المساحة ديناميكياً باستخدام ALTER SYSTEM كما عو الحال مع ال Shared Pool وغيرها.

5- تعمل ال Large Pool فقط في بيئة ال Shared Server.

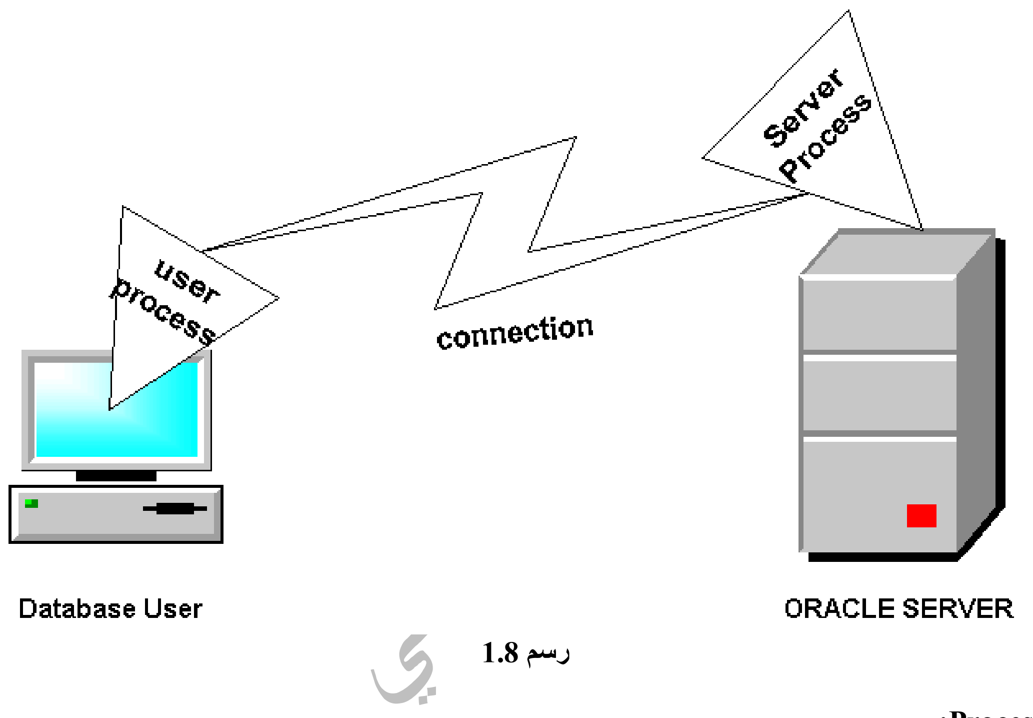
سوف يتم التطرق لاحقاً ل Shared Server.

:JAVA POOL

هي منطقة ذاكرة اختيارية اخرى توفر ذاكرة لعمليات ال Java المختلفة وتعمل بنفس فكرة ال SQL أو PL/SQL في ال Shared Pool، مساحة ال Java Pool متحكمة من قبل العامل "Parameter" JAVA_POOL_SIZE.

CONNECTION

لكي تستطيع الدخول الى ال **Database** للقيام بعلميات مختلفة مثل استخراج بيانات عبر جمل ال **SQL** ، يجب أولاً ان يكون المستخدم متصلاً بال **Instance**.
يقوم المستخدم باستخدام أحد البرامج التي يمكن عبرها الاتصال بال **Instance** مثل **SQLPLUS** فينشئ ما يعرف باسم **User Process**. وبعد ان يتم الدخول الى ال **Oracle Server** ويتم الاتصال بال **Instance** ينشئ ما يعرف باسم **Server Process**. يقوم ال **Server Process** بعملية الوسيط بين المستخدم **User Process** وال **Instance** حيث يقوم بتنفيذ العمليات التي طلبها المستخدم مثل جمل ال **SQL**.



رسم 1.8

:Process

تعريف ال **Process** بشكل عام في الأوراكل هي العملية أو الوظيفة التي تنفذ مجموعة من الخطوات أو مهمات محددة.

يوجد ثلاث أنواع من ال **Processes** في الأوراكل هم: **User Process , Server Process, Background Process**.

سوف يتم التطرق لاحقاً ل Background Process.

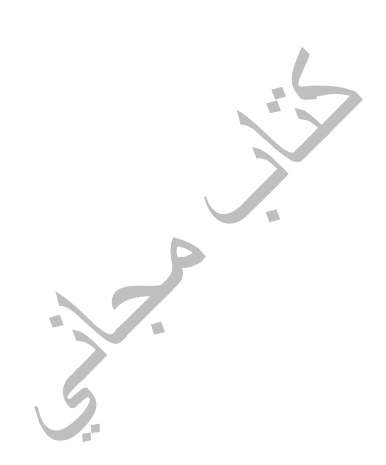
:Session

ال **Session** هو حالة الاتصال الموجودة بين المستخدم و ال **Oracle Server**. يبدأ ال **Session** عند دخول المستخدم ال **Oracle Server** وينتهي عند خروج المستخدم من ال **Oracle Server** سواء بإرادة المستخدم (عند انتهاء المهمة) أو لا ارادياً (عند حدوث عطل ما).
يمكن للمستخدم الواحد اين يكوّن أكثر من **Session** واحد في نفس الوقت اذا استخدم أكثر من برنامج مثل **SQL PLUS** و **ORACLE FORMS** عدا بعض الحالات القليلة.

:PROCESSING A STATEMENT (DML)

الخطوات التي تمر عبرها جمل ال SQL (جمل ال DML) ملخصة في الخطوات التالية:

- **يتم الاتصال بال Instance وتكوين User Process و Server Process.**
- **اذا كانت البيانات المطلوبة غير متوفرة في ال Database Buffer Cache فيقوم ال Server Process باحضار البيانات من Data Files.**
- **يقوم ال Server Process بوضع قيود "Locks" على البيانات التي يتم تعديلها، ويتم تخزين البيانات السابقة (قبل التعديل) في منطقة تسمى ROLLBACK (SEGMENT) BLOCK حتى يستطيع المستخدم باسترجاع البيانات الأصلية غير المعدلة اذا تم اختيار Rollback عوضاً عن Commit.**
- **اذا تم اختيار Commit فإن البيانات المعدلة يتم نقلها الى Data Files .**



PROGRAM GLOBAL AREA (PGA)

هي منطقة ذاكرة غير مشتركة و خاصة لعملية واحدة فقط "Process ". تخصص ذاكرة ال PGA لكل عملية "Process " عند بداية العملية "Process " و تزال ذاكرة ال PGA عند نهاية العملية "Process" . وتحتوي ذاكرة ال PGA على البيانات الخاصة لتلك العملية "Process"، مكونات ال PGA تعتمد على تعريف ال Server حيث يوجد تعريفان هما Dedicated وShared.
في نظام ال Dedicated Server يتكون Server Process واحد خاص لطلبات User Process واحد (مستخدم واحد)، أما في ال Shared Server فيتم المشاركة في ال Server Process من قبل أكثر من User Process (يمكن ان يكون هناك أكثر من Server Processes و ليس بالضرورة واحد فقط أي مثلاً يمكن ان يوجد خمسة Server Processes يتشارك بهم خمسين مستخدم User Processes)

بعض مكونات ال PGA:

- **SORT AREA: حيث تخصص لأي طلب من قبل ال Process يحتوي على الأوامر التالية: Group By, Order By , Distinct أو ال Set Operators مثل Minus, Union, Intersect.**
- **SESSION INFORMATION: حيث تخصص لمعلومات المستخدم مثل ال User Privileges.**

BACKGROUND PROCESSES

تكون ال Background Processes متوفرة وجاهزة للعمل بعد تشغيل ال Instance.
كل Background Process مسؤول عن مهمات خاصة، ويوجد نوعيين من Background Processes هما أختياري و إلزامي (أو إجباري).

تذكر: الرسم رقم 1.1 ومكان وجود ال Background Processes.

أمثلة:

- Database Writer (DBWn)
- Log Writer (LGWR)
- System Monitor (SMON)
- Process Monitor (PMON)
- Checkpoint (CKPT)
- Archiver (ARCn)

:DATABASE WRITER (DBWn)

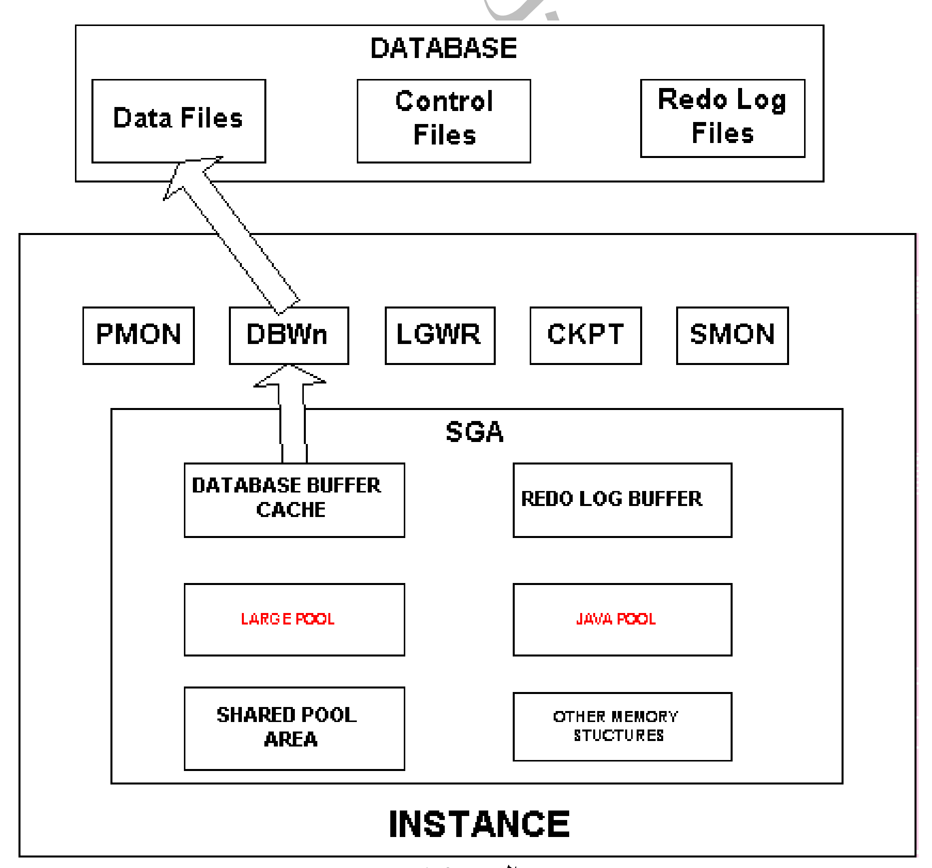
يقوم بكتابة البيانات المتغيرة "Dirty Buffers" من ال Database Buffer Cache الى ال Data Files كما هو موضح بالرسم 1.9، وبذلك يتم تفرغة Buffers باستمرار من أجل البيانات المتغيرة اللاحقة.

تذكر: أن البيانات المخزنة في ال Instance تسمى Buffers والبيانات المخزنة في ال Data Files تسمى Blocks.

وتعمل ال DBWn عند حدوث أحد الأحداث التالية:

- عند حدوث Checkpoint.
- عدد ال Dirty Buffers (أو ال Buffers الممتلئة بالبيانات المتغيرة) يصل الى الحد الأعلى المسموح فيه.
- عند عدم وجود من "Free Buffers" فارغة بعد البحث عنها من قبل ال Process .
- عند وضع Tablespace سواء من النوع العادي أو الموؤقت في حالة الإغلاق "Offline".
- عند وضع Tablespace في حالة القراءة فقط (اي لا يمكن تغيير البيانات بل فقط الاطلاع عليها)"Read Only".
- عند وضع Tablespace في حالة ال Backup.
- عند استخدام أوامر Drop و Truncate الخاصة بال Table.
- عند حدوث Timeout.

سوف يتم التطرق لاحقاً ل Checkpoint و ال Tablespace.



الرسم 1.9

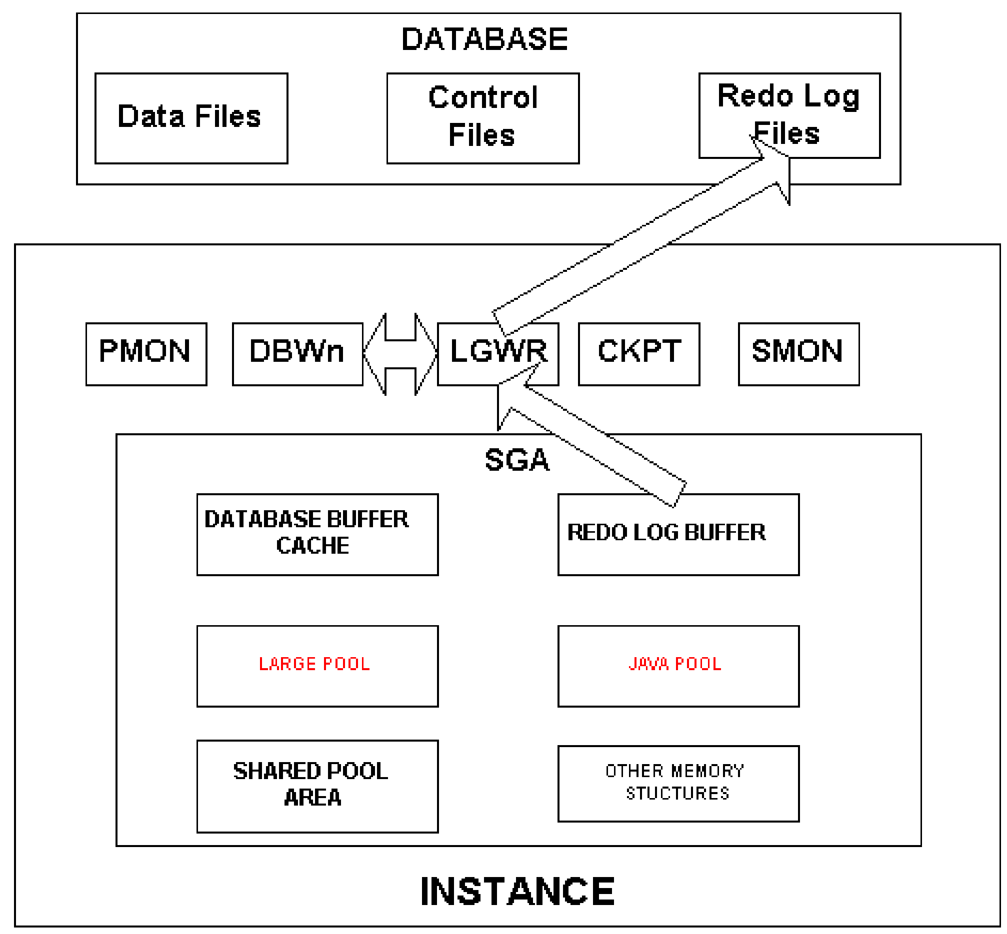
:LOG WRITER (LGWR)

يقوم بنقل البيانات المخزنة في ال Log Buffer Cache الى ال Redo Log Files بشكل دوري مما يساعد على اخلاء ذاكرة لل Redo Entries الجديدة ، كما هو موضح بالرسم 1.10.

تذكر: أن البيانات المخزنة في ال Redo Log Buffer تسمى Redo Entries.

وتعمل ال LGWR عند حدوث أحد الأحداث التالية:

- عند امتلاء ثلث ال Redo Log Buffer.
- عند وجود بيانات في ال Redo Log Buffer حجمها اكبر من 1 MB.
- قبل ان تقوم ال DBWn بنقل البيانات من ال Database Buffer Cache الى ال Data Files .
- كل 3 ثوان.
- عند Transaction Commit.



الرسم 1.10

:SYSTEM MONITOR (SMON)

في حالة حدوث اي عطل مفاجئ لل Instance جميع البيانات الموجودة في ال Instance (مثل بيانات ال Database Buffer Cache) التي لم يتم نقلها الى ال Data Files على القرص الصلب تضيع أو تمسح. بعد هذه الحالة من ضياع بيانات ال Instance يعمل ال SMON بشكل اتوماتيكي لاسترجاع بيانات ال Instance و تسمى العملية ب Instance Recovery.

ويقوم ال SMON بتأدية عدد من الوظائف هي:

- **استرجاع البيانات المتغيرة التي تم تسجيل ما طرأ عليها من تغير في ال Redo Log Files ولكن لم يتم تسجيلها في ال Data Files. حيث يقوم ال SMON بقراءة ال Redo Log Files و معرفة التغيرات التي طرأت على البيانات و تغيير البيانات في ال Data Files.**
- **اعادة تجهيز ال Database ليتمكن المستخدم من الدخول مجدداً .**
- **الغاء جميع ال Transactions التي لم يحدث لها Commit وازالة القيود عن البيانات " Data Lock" .**
- **تحرير ال Temporary Segments واستعادة مساحتها الى ال Data Files.**
- **جمع المساحات الخالية الصغيرة الضائعة التي تنشئ بين البيانات نتيجة للتغيرات التي تحدث و تعيدها الى المساحة الخالية الرئيسية.**

تذكر: أن ال LGWR يعمل قبل ال DBWn.

سوف يتم التطرق لاحقاً في الفصول التالية لل Temporary Segment.

:PROCESS MONITOR (PMON)

في حالة حدوث أي عطل لل Processes قد يؤدي الى مشاكل داخلية في ال Database فيقوم ال PMON بتنظيف أخطاء ال Processes.

ويقوم ال PMON بتأدية عدد من الوظائف هي:

- **تحرير كل المصادر وال Locks على ال Tables أو ال Rows التي وضعها ال Process الفاشل.**
- **اعادة (Rollback) ال Transaction الخاص بالمستخدم (اي الغاء اي تعديلات لم يتم حدوث Commit لها)**

:CHECKPOINT (CKPT)

ال **Checkpoint** هي عملية كتابة جميع البيانات المتغيرة في **Database Buffer Cache** الى ال **Data Files** عبر ال **DBWn**. من فوائد ال **Checkpoint** ان البيانات التي تتغير باستمرار تنقل بشكل نظامي و دوري الى ال **Data Files** وبذلك يتم حفظ البيانات بشكل أمن.
ال **Checkpoint Process** يقوم بتجديد **"Update"** ال **Data Files Header** و ال **Control Files**.

من البيانات التي تجدد **"Update"** :

- رقم ال **Checkpoint** في ال **Data Files Header**.
- رقم ال **Checkpoint** في ال **Control Files**.
- رقم ال **Log Sequence** في ال **Control Files**.
- ال **SCN** في ال **Control Files**.
- اسماء ال **Archived Log**.

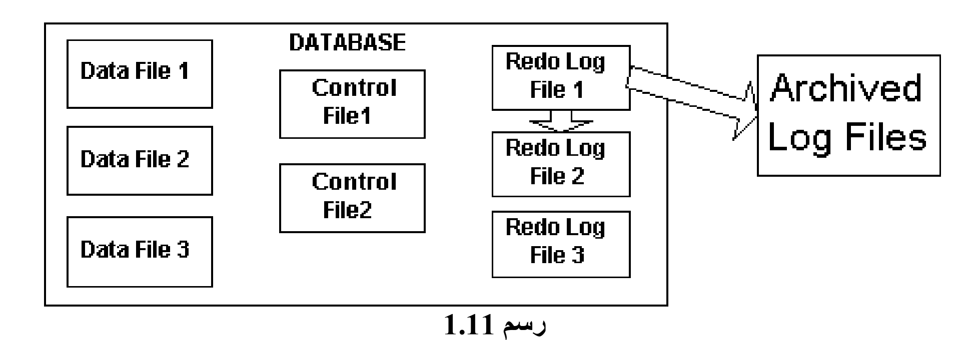
سوف يتم التطرق لاحقاً في الفصول التالية لل Data Files Header و ال Log Sequence وال SCN واسماء ال Archived Log.

:ARCHIVER (ARCn)

هو **Process** اختياري و ليس اجباري ووظيفته تكمن في نقل بيانات ال **Redo Log Files** الى **Archived Log Files**. باعتباره اختياري فله حالتيين إما يعمل أو مغلق. و لوضعه في وضيعة العمل يجب أن يكون ال **Database** في وضعية ال **ARCHIVELOG**، وفي حال تم وضع ال **Database** تحت هذه الوضعية فإن **ARCn** يعمل.
اذاً هناك حالتيين لوضعية ال **Database** تحددان عمل أو ايقاف ال **ARCn** هما:

:Archivelog

عند امتلاء الملف الأول من ملفات ال **Redo Log Files** يتم البدأ في كتابة البيانات في الملف الثاني وتسمى هذه العملية **Log Switch** ، ثم يقوم ال **ARCn** بشكل أوتوماتيكي بنقل بيانات الملف الأول الى ال **Archived Log Files**.



رسم 1.11

: NoArchivelog

عند امتلاء جميع ملفات ال **Redo Log Files** يتم اعادة كتابة البيانات **"Overwrite"** في الملف الأول على البيانات السابقة ، ولكن لا يتم كتابة البيانات فوق البيانات السابقة قبل ان يحدث **Checkpoint** للملف أو مجموعة الملفات الممتلئة.

سوف يتم التطرق لاحقاً لكيفية وضع ال Database تحت حالة ال Archivelog أو حال Noarchivelog.

الفصل الثاني

تنصيب و إدارة الأوراكل

INSTALLING & MANAGING ORACLE

DATABASE ADMINISTRATION TOOLS

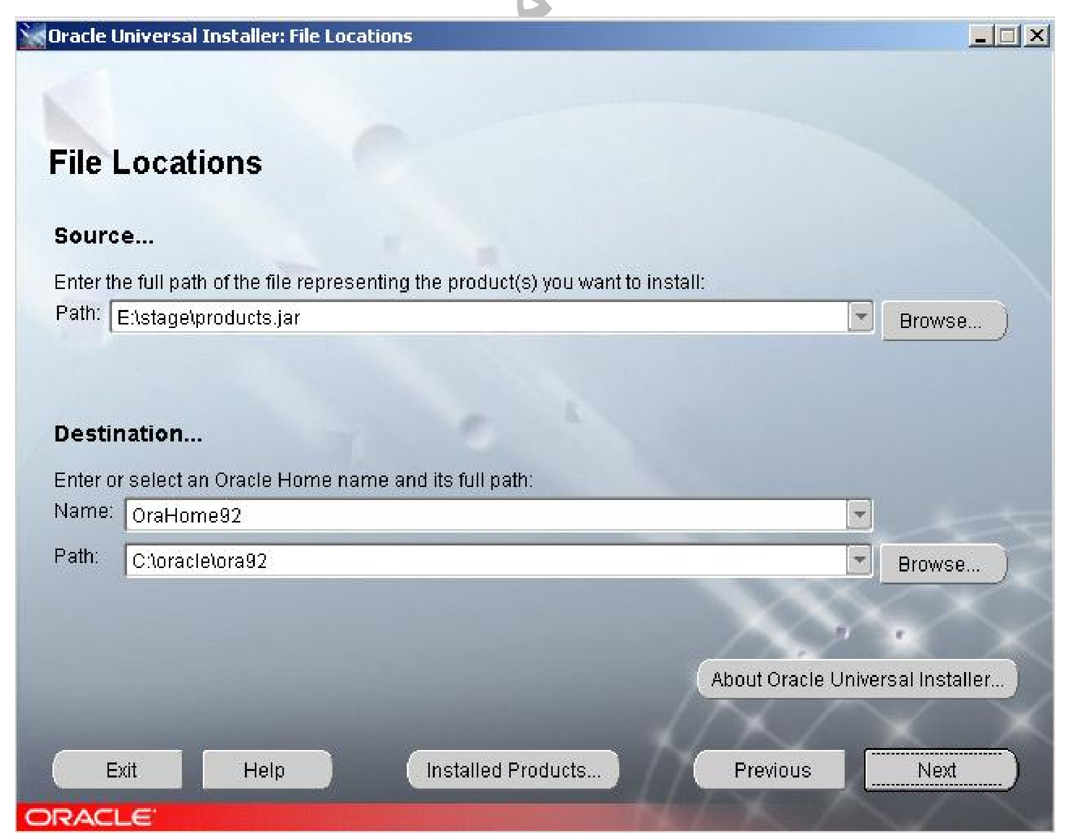
يتوفر عدد من الأدوات "Tools" في الاوراكل أهمها :

- ORACLE UNIVERSAL INSTALLER (OUI)
- ORACLE DATABASE CONFIGURATION ASSISTANT
- SQL PLUS
- ORACLE ENTERPRISE MANAGER (OEM)
- PASSWORD FILE UTILITY

:ORACLE UNIVERSAL INSTALLER (OUI)

يمكن استخدام ال OUI لتنصيب "Install" الاوراكل. في نظام الويندوز "Windows" تستطيع بدأ ال OUI بالضغط على ملف التنصيب ال Setup.exe ، أما في نظام ال Unix فيجب كتابة runInstaller.

الرسم 2.1 يوضح ال OUI في نظام الويندوز "Windows".



الرسم 2.1

من أهم مميزات ال OUI:

- يمكن تنصيب الأوراكل اكثر من مرة على ذات موقع التخزين مثل الHard Disk باستخدام اسم مخلتف لل ORACLE HOME الموضح في رسم 2.1 (أسفل كلمة Destination) و يمكن تنصيب نسخة قديمة "Old Version" للاوراكل مع نسخة جديدة "New Version" في ذات الموقع.

- يمكن تنصيب الأوراكل من الإنترنت عبر وصلة ال HTTP حيث تواجد نسخة الأوراكل، وبذلك يمكن تنصيب الأوراكل من قبل فرع شركة في دولة غير الدولة التي يوجد بها مقرها الأصلي (حيث تتواجد نسخة الأوراكل).

- يمكن أن يستعمل لإزالة الأوراكل من الجهاز أو السيرفر (Uninstall or Deinstall)

- يعمل بعدد مختلف من اللغات مثل الانجليزي و الفرنسي و الألماني و غيرها.

- يمكن ان يعمل تحت الأنظمة التي لا توفر نظام ال GUI والتي تكون فيها الأوامر بشكل كتابي Command وليس كما في ال Windows الذي يعمل تحت نظام ال GUI مثل الرسم 2.1. وتسمى الطريقة باسم الطريقة الصامتة "Silent Mode" ويتم استعمال ملف يسمى **Response File**.

<u>Response File:</u>

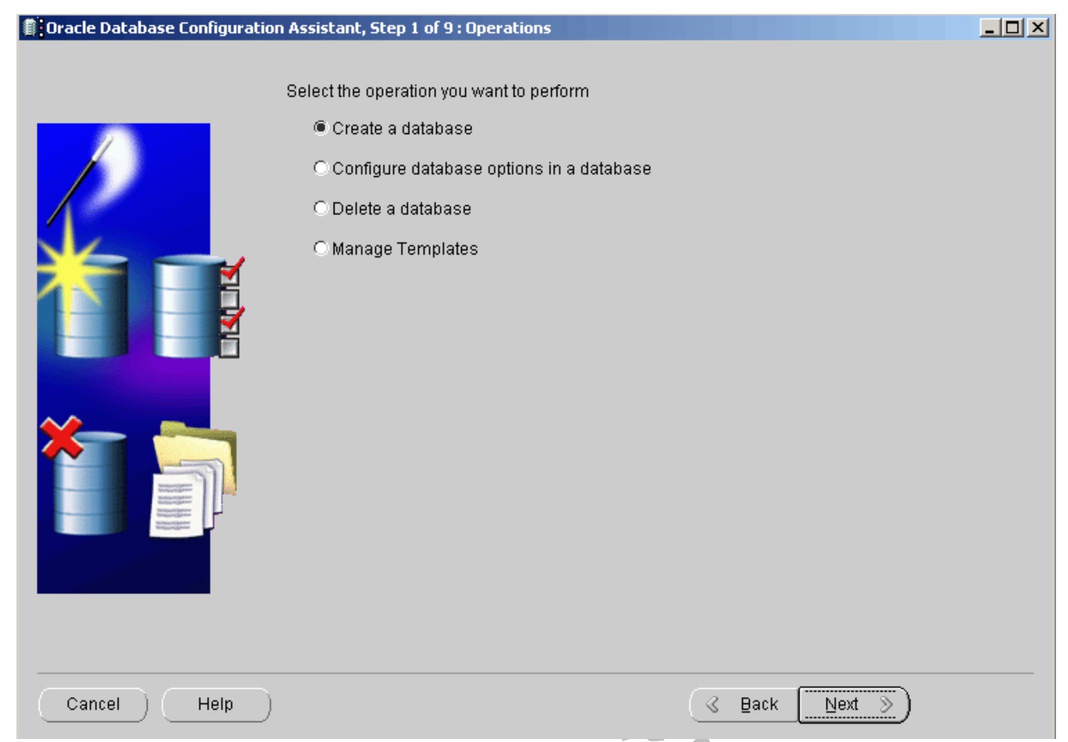
هو ملف (any name.txt) يحفظ البيانات اللازمة لعملية التنصيب مثل البيانات المطلوبة في الرسم 2.1 (مثل بيانات ال Source و ال Destination).

ORACLE DATABASE CONFIGURATION ASSISTANT:

يستخدم للأغراض التالية (انظر الرسم 2.2):

- تكوين البيانات "Database".
- حذف البيانات "Database".
- تعديل خواص البيانات "Database".
- ادارة ال Templates.

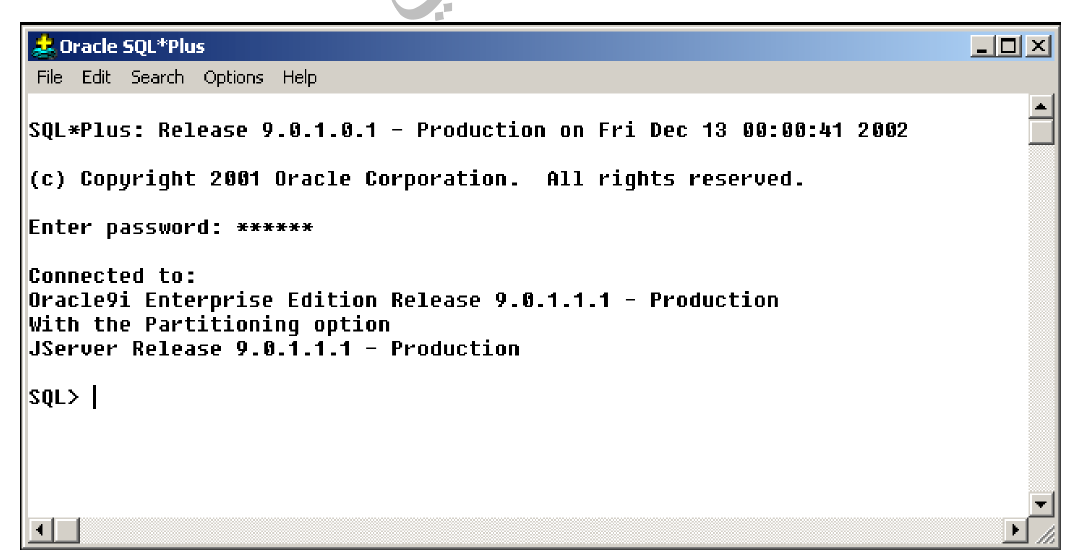
سوف يتم التطرق لاحقاً في فصل أخر لجميع الأغراض.



رسم 2.2

:SQL PLUS

هي اداة تمكن المستخدم من التفاعل مع البيانات، حيث تمكن المستخدم من الدخول الى البيانات، استخراج بيانات، اضافة بيانات، تعديل بيانات، واغلاق البيانات "Database". من المفروض أن تكون قد عملت مع ال SQL PLUS خلال دراستك لل SQL وتعرفت عليه جيداً.



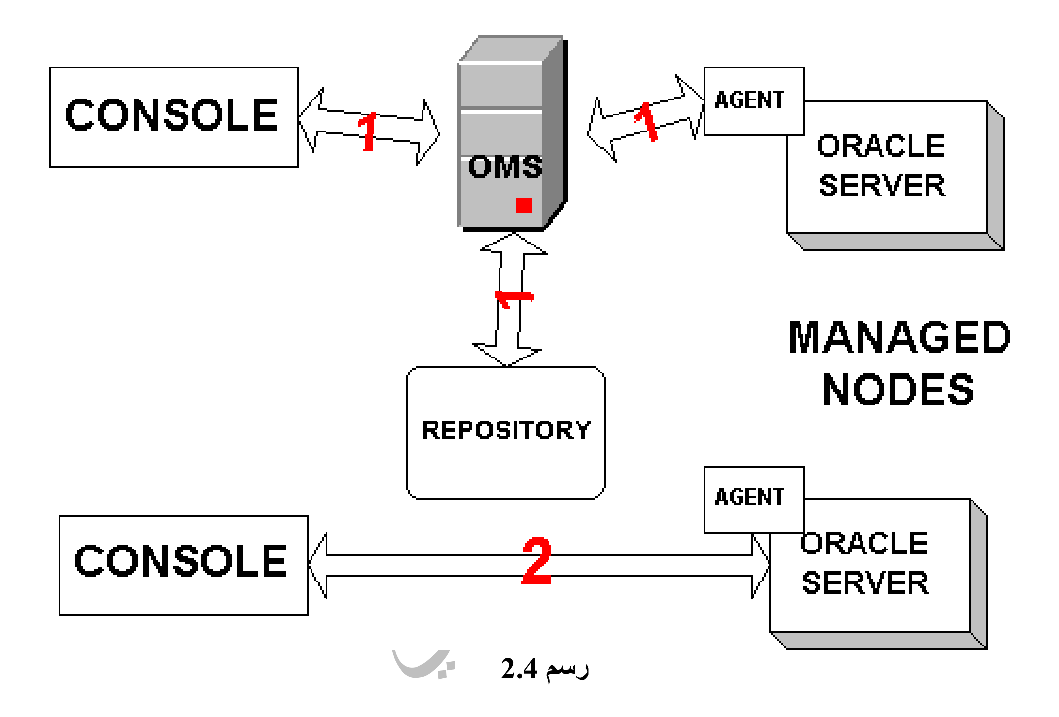
رسم 2.3

:ORACLE ENTERPRISE MANAGER (OEM)

هو نظام أداري يعمل على التحكم بمكونات الأوراكل وادارة البيانات من خلال نظام ال GUI(اي يتم القيام بمعظم الأوامر من خلال استعمال الفأرة "Mouse").

من أهم مكونات ال OEM:

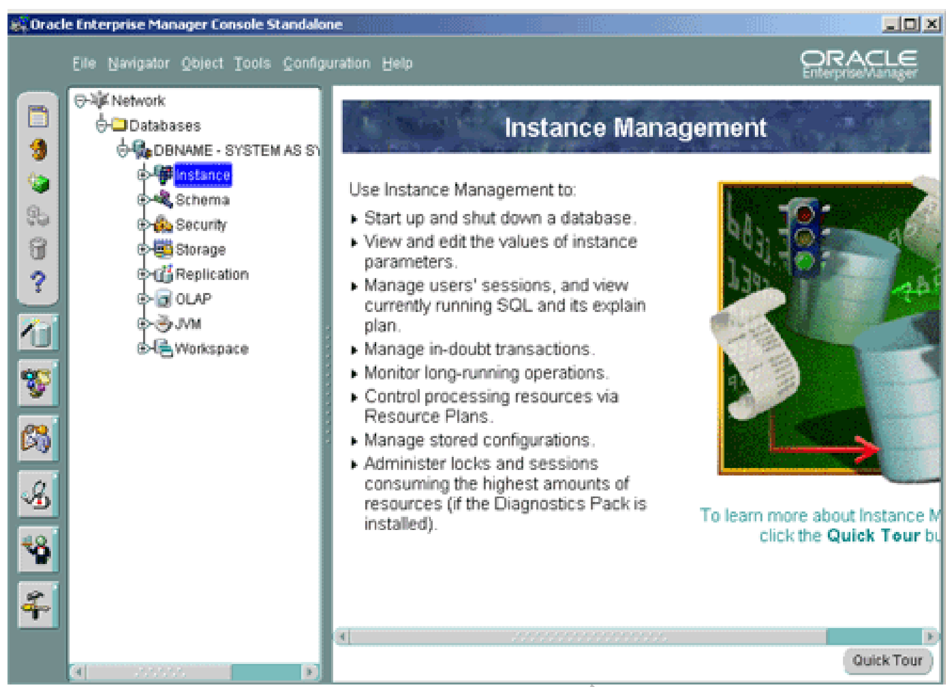
- CONSOLE
- ORACLE MANAGEMENT SERVER
- MANAGED NODES



رسم 2.4

:Console

أداة "GUI Tool" من الأدوات الهامة حيث تعمل كمحطة لإدارة جميع المهمات في أوراكل. من هنا يمكن التحكم بالInstance ويمكن التحكم بال Tables و غيرها و تنفيذ المهمات عن طريق الفأرة "Mouse" والإختيارات المتعددة عوضاً عن تنفيذ المهمات بواسطة جمل ال SQL.



رسم 2.5

من الملاحظ من الرسم 2.4 أن ال Console يستطيع الدخول الى Oracle Server بطريقتيين هما:
Standalone: بطريقة مباشرة دون الحاجة الى Oracle Management Server. (سهم رقم 2)
عبر ال OMS: عبر ال Oracle Management Server. (أسهم رقم 1)

DBA Tools:

هي مجموعة من البرامج التي يمكن تشغيلها من ال Console:

Instance Manager: يستخدم لتشغيل واغلاق ومراقبة البيانات "Database".
Security Manager: يستخدم لإدارة Users Accounts و Privileges.
Storage Manager: يستخدم للتحكم في ال Tablespaces وال Data Files وغيرها.
Schema Manager: يستخدم لتكوين وادارة ال Tables و Views وال Indexes.
SQL PLUS Worksheet: اداة لكتابة جمل ال SQL وتنفيذها.

Oracle Management Server (OMS):

أهم مكونات ال Oracle Enterprise Manager يقوم بدور الوسيط بين ال Console و ال Managed Nodes في تبادل البيانات بالاضافة الى ادارة User Accounts وعمليات مختلفة مثل Jobs, Events.
يقوم باستعمال ال Repository لتخزين بيانات النظام "System Data" وبيانات البرامج والأدوات و بيانات ال Managed Nodes.

:Managed Nodes

قد تتكون ال Node من ال Database وخدمات أخرى. يوجد على كل Node ما يعرف باسم Oracle Intelligent Agent يقوم بالتواصل مع OMS ويؤدي المهام المرسلة من قبل ال Console، وهو يعمل بشكل منفرد ويمكن ان يؤدي مهام مثل اغلاق و تشغيل ال Database ويلزم Oracle Intelligent Agent واحد فقط على كل Node.

:PASSWORD FILE UTILITY

تعرف باسم Orapwd وتستخدم في تكوين ال Password File.

لتكوين ال Password File يجب اتباع القاعدة التالية:

```
$ORAPWD FILE = filename PASSWORD = yourpassword ENTRIES=number;
```

حيث ان ال File هو اسم الملف وموقعه على الجهاز، وال Password هي كلمة السر الخاصة ب SYSDBA و SYSOPER، وال Entries تمثل العدد الأقصى من المستخدميين الذين يحق لهم الدخول الى ال Database في هيئة SYSDBA او SYSOPER.

عندما تقوم بالدخول كمستخدم ال SYSDBA فانك تدخل الى SYS Schema وليس ال Schema الخاص بك وكذلك عند الدخول كمستخدم ال SYSOPER فانك تدخل الى PUBLIC Schema.

تذكر: ملف ال Password File من الفصل الأول.

AUTHENTICATION METHODS

للقيام بالمهمات الإدارية للبيانات "Database" يقوم الأوراكل بتكوين اثنين User Accounts عند بداية تكوين البيانات "Database" هما SYS و ال SYSTEM الذين يملكان جميع ال Privileges في الأوراكل أو ما يسمى ال DBA Role.

ملاحظة: كلمة السر لل SYS عند بداية تكوين البيانات هي change_on_install اما كلمة السر لل SYSTEM هي manager.

ملاحظة2: يعتبر ال SYS هو المالك "Owner" ل Database Data Dictionary.

هناك طريقتين يمكن السماح فيها للمستخدمين الذين يملكون ال DBA Role الدخول الى ال Database وادارة البيانات هما: Password File Authentication وOperating System Authentication.

:PASSWORD FILE AUTHENTICATION

لقد تطرقنا قبل قليل لل Password File Utility والتي تستخدم في هذا النظام، ولتشغيل هذا النظام يجب اتباع الخطوات التالية:

1- تكوين ملف ال Password File عن طريق ال ORAPWD.

2- وضع ال REMOTE_LOGIN_PASSWORDFILE الذي هو أحد مكونات ال Initialization Parameter File الى الحالة EXCLUSIVE.

3- اعطاء "Grant" ال Privileges الخاص ب SYSDBA أو SYSOPER الى المستخدمين الجدد الذين تم اختارهم للحصول ال DBA Role.

مثال تطبيقي 2.1:

الخطوة 1: اكتب التالي في ال SQL PLUS

```
$ORAPWD FILE=c:\oracle_home\ora92\dbs\myfile PASSWORD=newadmin
ENTRIES=5;
```

الخطوة 2:

```
SHOW PARAMETER REMOTE_LOGIN_PASSWORDFILE;
```

اذا كانت الحالة غير EXCLUSIVE فيجب تغيير الحالة في ال Initialization Parameter File.

الخطوة 3:

```
GRANT SYSDBA TO Ahmad;
```

للدخول الى ال Database اكتب التالي:

```
CONNECT sys/newadmin AS SYSDBA;
```

:OPERATING SYSTEM AUTHENTICATION

بمجرد الدخول الى ال Operating System (مثال: الويندوز) تستطيع الدخول الى ال Database دون الحاجة الى كلمة سر أو اسم مستخدم.

ملاحظة: يجب أن يكون ال REMOTE_LOGIN_PASSWORDFILE في حالة NONE ليعمل النظام.

مثال تطبيقي 2.2:

بعد الدخول الى ال SQLPLUS اكتب التالي:

```
CONNECT / AS SYSDBA;
```

حالات ال Remote_Login_Passwordfile:

يوجد ثلاثة حالات يمكن وضع ال Remote_Login_Passwordfile فيها هي:

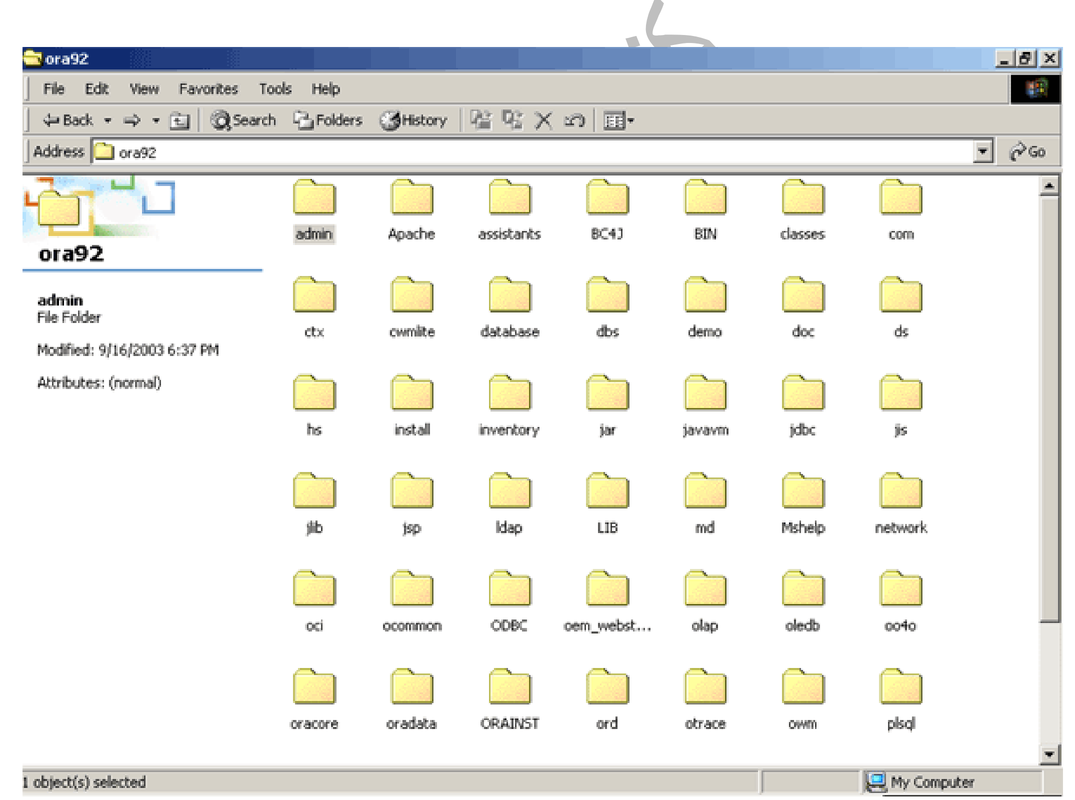
- **EXCLUSIVE: تحدد ان Instance واحد فقط يمكن ان يستخدم ال Password File ويمكن اعطاء "Grant" ال SYSDBA Privileges أو ال SYSOPER Privileges لمستخدمين أخرين، وهي ضرورية لنظام ال Password File Authentication.**
- **SHARED: تحدد ان باستطاعة أكثر من Instance استخدام ال Password File ولكن لا يمكن اضافة مستخدمين أخريين لل SYSDBA أو SYSOPER والمستخدم الوحيد المعرف في الملف هو SYS.**
- **NONE: ضروري لنظام Operating System Authentication.**

OPTIMAL FLEXIBLE ARCHITECTURE

يعتبر ال **OFA** نظام لترتيب و توزيع ملفات ال **Database** المختلفة على حسب نوعها أو استخدامها، وهو معتمد في جميع الأنظمة التي يعمل بها الأوراكل مثل **Windows, UNIX**.
باستخدام نظام ال **OFA** يمكن ان نحقق عدة فوائد منها سهولة التمييز بين الملفات المختلفة وسهولة ايجادها وسهولة ادارة الأوراكل بتوزيع الملفات كلّ حسب نوعه، وتسهيل التحكم في التوسع الذي يطرأ على ال **Database** في المستقبل.

و باستخدام نظام ال**OFA** نستطيع:

- اعتماد اسلوب ثابت لتسمية الملفات لكي نستطيع ايجاد الملفات بسهولة.
- تفريق ملفات ال **Oracle Software** عن **Oracle Database**.
- تفريق ملفات النسخ المختلفة **"Versions"** من الأوراكل.
- تفريق ملفات ال **Data files** عن ملفات ال **Control Files** عن ملفات ال **Redo Log Files**.



رسم **2.6**

كما تلاحظ توزيع ملفات ال **Database** المختلفة الى مجلدات مختلفة. فمثلاً تجد ملف ال **Password File** تحت مجلد ال **dbs** بالمقابل تجد ملفات ال **Control Files** تحت مجلد **oradata** و هكذا...

INITIALIZATION PARAMETER FILE

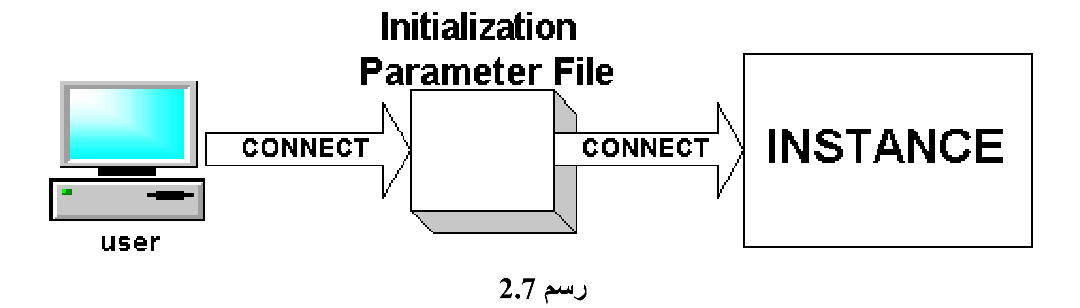
لتشغيل ال Instance يجب توفر ال Initialization Parameter File. يمكن استعمال أكثر من ملف لنفس ال Database. هنالك نوعان من ال Initialization Parameter File هما:

- PFILE: يطلق عليه اسم الملف الثابت "Static File" ويعرف باسم initSID.ora
- SPFILE: يطلق عليه اسم الملف المقاوم "Persistent File" ويعرف باسم spfileSID.ora

ملاحظة: ال SID عبارة عن رمز لإسم ال Instance. فاذا كان اسم ال Instance "DbB" فيكون اسم الملف (initDbB.ora).

بعض محتويات ال Initialization Parameter File:

- العوامل "Parameters" الخاصة بال Instance وال SGA مثل ال SGA_MAX_SIZE.
- اسم ال Database التي يعمل عليه ال Instance.
- اسماء و مواقع ال Control Files.
- بيانات عن ال Undo Segment.
- تحديد إما Archivelog أو Noarchivelog.



رسم 2.7

PFILE:

هو ملف من نوعية ال TXT الذي يمكن فتحه وتغييره بواسطة ال Notepad وغيرها من برامج الكتابة. يتم قراءة محتوياته عند بداية تشغيل ال Instance ومعظم العوامل "Parameters" الموجودة داخله لا تعمل بشكل ديناميكي (يستلزم اغلاق وفتح ال Instance من جديد لكي يحدث التغيير) عدا بعض العوامل "Parameters" التي يمكن تعدل ديناميكياً في الذاكرة ولكن تظل في ال PFILE كما هي (أي انه عند تشغيل ال Instance مرة أخرى يرجع الوضع الى قبل التعديل الديناميكي).
يوفر ال OUI ملف مبدئي عند بداية تكوين ال Database اسمه init.ora يمكن استعماله لتكوين ال PFILE.

بعض مكونات ال PFILE:

```
db_block_size=4096
db_cache_size=33554432
open_cursors=300
background_dump_dest=C:\oracle2\ora92\admin\dbname\bdump
core_dump_dest=C:\oracle2\ora92\admin\dbname\cdump
timed_statistics=TRUE
user_dump_dest=C:\oracle2\ora92\admin\dbname\udump
db_domain=oracle
remote_login_passwordfile=EXCLUSIVE
control_files=("C:\oracle2\ora92\oradata\dbname\CONTROL01.CTL",
"C:\oracle2\ora92\oradata\dbname\CONTROL02.CTL",
"C:\oracle2\ora92\oradata\dbname\CONTROL03.CTL")
compatible=9.0.0
db_name=dbname
instance_name=dbname
java_pool_size=33554432
large_pool_size=1048576
shared_pool_size=33554432
processes=150
fast_start_mttr_target=300
sort_area_size=524288
undo_management=AUTO
undo_tablespace=UNDOTBS
```

قواعد ال PFILE:

- كل العوامل "Parameters" اختيارية، أي اذا لم يحدد ال DBA قيمة العوامل "Parameters" أو لم يتم كتابة العامل "Parameter" في ملف ال PFILE يقوم الأوراكل بتحديد العامل "Parameter" و تكون قيمته هي القيمة المحددة سلفاً من قبل الأوراكل "Default Value"
- ترتيب العوامل "Parameters" في ال PFILE غير مهم.
- اذا كان هناك أكثر من قيمة للعامل "Parameter" الواحد فيجب وضع القيم داخل قوسين و تفريقهم بفاصلة كما في Control_Files.
- تحدد قيمة العامل "Paramter" بهذه الصيغة (العامل = القيمة).
- يمكن استعمال (#) في الملف لوضع ملاحظات داخل الملف (هذه البيانات لل DBA ولا يقرأها الأوراكل).

:SPFILE

هو ملف من نوعية ال binary الذي يقوم بادارته ال Oracle Server(فتحه و تعديله) ويمكن تعديل جميع قيم العوامل "Parameters" الموجودة داخله بشكل ديناميكي (اي باستخدام ال ALTER SYSTEM) بحيث يتم تعديل بيانات ال SPFILE في الذاكرة أو الملف أو الأثنين معاً (أي انه عند تشغيل ال Instance مرة أخرى تكون التغيرات التي اجريت على ال SPFILE بواسطة ال ALTER SYSTEM موجودة وثابتة).
لا يقوم الأوراكل بتكوين ملف SPFILE ابتدائي كما يحدث مع ال PFILE ولكن يتم تكوين ال SPFILE من ال PFILE كا سيأتي شرحه بعد قليل.

لتعديل العوامل "Parameters" داخل SPFILE يجب اتباع القاعدة التالية:

```
ALTER SYSTEM SET PARAMETER = VALUE [SCOPE = SCOPE_VALUE];
```

ملاحظة: استخدام ال [ ] في القواعد يدل على أن هذا الجزء اختياري و في حالة لم يتم كتابته يختار الأوراكل ال Default . وال Default يكون أحد الحلات الأكثر شيوعاً وهي تم اختيارها مسبقاً من قبل مبرمجي الأوراكل.

أما حالات ال SCOPE_VALUE ثلاث هي:

- **MEMORY: يتم تغيير قيمة "Value" ال Parameter فقط في الذاكرة، أي عند تشغيل ال Instance مرة أخرى تلغى التغيرات.**
- **SPFILE: يتم تغيير قيمة "Value" ال Parameter في ملف ال SPFILE فقط.**
- **BOTH: يتم تغيير قيمة "Value" للعامل "Parameter" في ملف ال SPFILE والذاكرة معاً.**

ملاحظة: اذا لم يتم كتابة الجزء SCOPE=SCOPE_VALUE فيعتبر الأوراكل انها SCOPE=BOTH (BOTH هي ال Default).

مثال تطبيقي 2.3:

```
ALTER SYSTEM SET SHARED_POOL_SIZE = 33554432    SCOPE=SPFILE;
```

بعض مكونات ال SPFILE:

```
*. background_dump_dest='C:\oracle2\ora92\admin\dbname\bdump'
*. compatible='9.0.0' *.db_block_size=4096 *.db_cache_size=33554432
*. db_domain='oracle'
………………
```

ليست كل العوامل "Parameters" في ال SPFILE اختيارية، يوجد سبع عوامل "Parameters" **يجب** ان تحدد هي:

- .BACKGROUND_DUMP_DEST
- USER_DUMP_DEST
- DB_NAME
- SHARED_POOL_SIZE
- COMPATIBLE
- DB_BLOCK_BUFFERS
- CONTROL_FILES

سوف يتم التطرق لاحقاً للعوامل "Parameters" الجديدة لاحقاً.

:Create SPFILE

يمكن تكوين ال SPFILE من ال PFILE اذا كان المستخدم يملك SYSDBA ROLE أو SYSOPER Role باتباع القاعدة التالية:

```
CREATE SPFILE [='location and name'] FROM PFILE[='location and name'];
```

يمكن اهمال ذكر اسم و موقع ملف ال SPFILE في القاعدة اذ يمكن للأوراكل تكوين الأسم على النمط المذكور سابقاً (spfileSID.ora) وفي الموقع ال Default وهو ضمن المجلد dbs. كذلك الحال بالنسبة لل PFILE حيث يختار الأوراكل الملف المعرف بالأسم initSID.ora وضمن المجلد dbs.

ملاحظة: قد يختلف الموقع ال Default من نسخة أوراكل الى أخرى. بعض نسخ الأوراكل تكَون الملف ضمن المجلد database.

مثال تطبيقي 2.4:

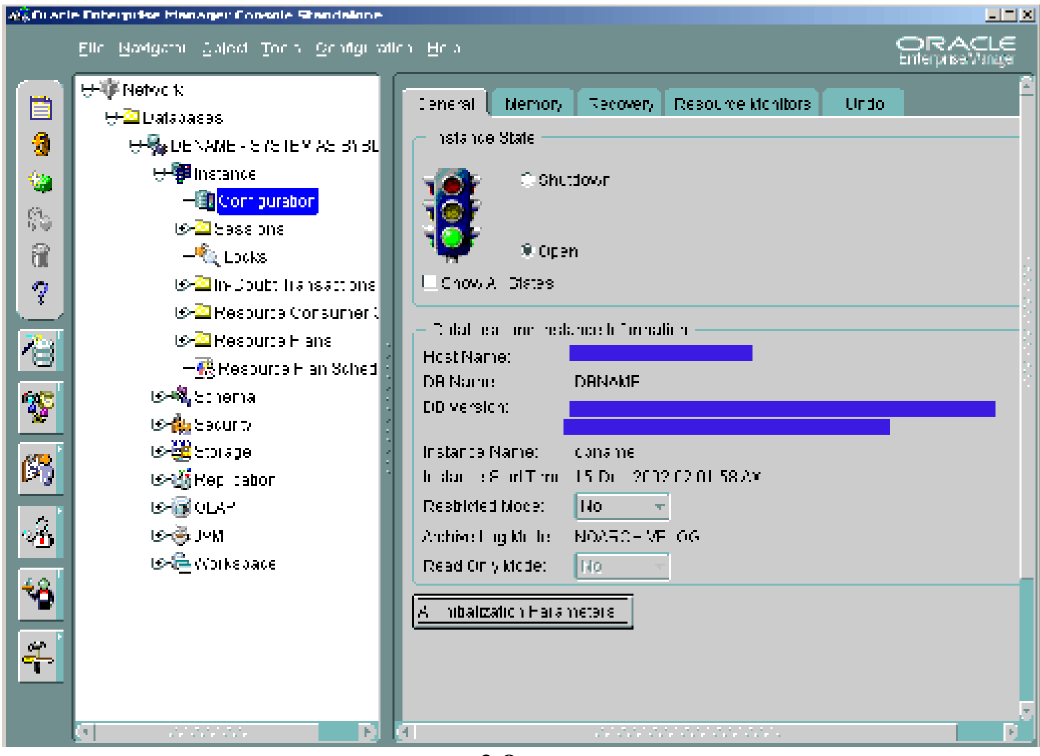
من ال SQL PLUS

```
CONNECT / AS SYSDBA;

CREATE SPFILE FROM PFILE;
```

يمكن تعديل ال SPFILE من ال CONSOLE باتخاذ الخطوات التالية:

- ادخل الى Enterprise Manager Console.
- اختر الوضع Standalone.
- اضغط على قائمة ال Database ثم قائمة "اسم ال Database"
- يطلب منك الاسم و كلة السر.
- اكتب اسم ال SYSTEM وكلمة السر الخاص به.
- تأكد من وضع Connect as الى الحالة SYSDBA.
- اضغط على قائمة ال Instance.
- اضغط على Configuration.
- اضغط على All Initialization Parameter.



رسم 2.8

START UP A DATABASE

يوجد ثلاث حالات يمكن أن نبدأ تشغيل ال Database ولكل منها خاصيته والوظائف التي يمكن اجرائها في تلك الحالة، الحالات الثالثة بالترتيب : NOMOUNT , MOUNT , OPEN.

:NOMOUNT

تسمى عملية Staring the Instance لأنه في هذه المرحلة فقط يبدأ تشغيل ال Instance دون باقي أجزاء ال Oracle Database.

يمكن القيام بوظيفتين في هذه الحالة هما:

- **تكوين ال Database عبر جملة ال SQL (CREATE DATABASE).**
- **تكوين أواعادة تكوين ملفات ال Control Files.**

سوف يتم التطرق لاحقاً لكيفية تكوين ال Database باستخدام جمل ال SQL.

يتم في هذه المرحلة عدة أمور هي:

- **تتم قراءة ملف ال Initialization Parameter File بحيث يبحث الأوراكل على ملف SPFILE إن لم يجد يبحث عن PFILE وإن لم يجد تتم قراءة الملف init.ora.**
- **حجز مساحة من الذاكرة لل SGA.**
- **بدأ ال Background Processes.**
- **فتح ملف يسمى alertSID.log وملفات تسمى ال Trace Files.**

سوف يتم التطرق لاحقاً لل alertSID.log و Trace Files.

ملاحظة: يجب تحديد قيمة العامل "Parameter" DB_NAME لكي يعمل ال NOMOUNT.

:MOUNT

في هذه المرحلة يمكن القيام بوظائف مهمة مثل:

- **تغيير "Rename" اسماء ال Data Files.**
- **امكانية وضع ال Database تحت وضعية ال ARCHIVELOG أو NOARCHIVELOG.**
- **عمل صيانة "Recovery" لل Database.**

يتم في هذه المرحلة عدة أمور هي:

- **فتح وقراءة ملفات ال Control Files.**
- **ربط ال Database بال Instance الذي سوف يعمل عليها.**

:OPEN

هي الحالة العادية التي تم العمل فيها بواسطة ال SQLPLUS ويتم اجراء تعديل وحذف واضافة بيانات وغيرها من المهام.

يتم في هذه المرحلة عدة أمور هي:

- **فتح ال Online Data Files .**
- **فتح ال Online Redo Log Files.**

في حالة فشل فتح ملفات ال Data Files وال Redo Files او في حالة حذفها أو نقلها من مواقعها المحددة في ال Control Files فلا يمكن ان تعمل الحالة OPEN.

ملاحظة: اسماء ومواقع ال Control Files مخزنة في ال Initialization Parameter Files اما اسماء و مواقع ال Data Files وال Redo Log Files فهي مخزنة في ال Control Files.

:STARTUP Command

لتشغيل ال Database في احدى الحلات المذكورة سابقاً تستعمل القاعدة التالية:

STARTUP [FORCE] [RESTRICT] [PFILE='location and name'] [OPEN [recover] [database's name] | MOUNT | NOMOUNT] ;

ملاحظة: استخدام ال | في القواعد يدل على انها (أو) "OR". فمثلاً لا يمكن لل MOUNT و ال OPEN ان يكتبا معاً.

أوامر القاعدة :

PFILE: سبق ذكر ان الأوراكل يبحث عن الملف عند بداية مرحلة ال NOMOUNT ولكن يمكن للمستخدم تحديد ملف محدد عوضاً على ان يقوم الأوراكل بالبحث عنه.

RECOVER: خاصة فقط بمرحلة ال OPEN اذ لا يمكن كتابتها مع ال MOUNT، وعند استخدامها يقوم الأوراكل بعمل عملية ال RECOVERY للبيانات.

RESTRICT: تحدد المستخدمين الذين يحق لهم الدخول الى ال Database وهم من يملك ال Restricted Session Privilege التي سوف يتم التطرق لها لاحقاً.

FORCE: اذا كان ال Instance يعمل في احد الحالات فيمكن اغلاقه ثم تشغيله مباشرة في الحالة العادية. اذاً هي مثل عملية ال "Restart" في الويندوز.

مثال تطبيقي 2.5:

لاستخدام قاعدة ال STARTUP يمكن القيام بالتالي:

1- ادخل الى ال SQLPLUS أو عبر ال Command Prompt.
2- ادخل اسم المستخدم و كلمة السر.
3- يجب الدخول الى SYSDBA عن طريق CONNECT \ AS SYSDBA;
4- اغلق ال Database بواسطة الجملة التالية: SHUTDOWN;

STARTUP; في هذه الجملة يعتبر الأوراكل انها OPEN باعتبارها ال Default.
STARTUP MOUNT;
STARTUP NOMOUNT;

ALTER DATABASE Command:

يمكن ان تستخدم في عدد من الوظائف منها :

1- تغيير حالة ال Database من :

- NOMOUNT الى MOUNT.
- MOUNT الى OPEN

يمكن استخدام أمر ال ALTER DATABASE باتباع القاعدة التالية:

ALTER DATABASE [database's name] MOUNT | OPEN;

مثال تطبيقي 2.6:

بينما يتم استخدام حالة ال NOMOUNT يمكن التحويل الى ال MOUNT بكتابة التالي:

ALTER DATABASE MOUNT;

2- لوضع ال Database في حالة القراءة فقط "Read-Only Database":

يمكن فتح ال Database في حالة OPEN بدون السماح بتغيير البيانات في ال Database باستخدام القاعدة التالية:

ALTER DATABASE [database's name] OPEN READ WRITE | READ ONLY;

مع العلم أن READ WRITE هي ال Default.

مثال تطبيقي 2.7:

بينما ال Database في الحالة العادية "READ WRITE" يمكن التحويل بكتابة الجملة التالية:

```
ALTER DATABASE OPEN READ ONLY;
```

من الوظائف التي يمكن القيام بها بينما ال Database في وضعية ال READ ONLY:

- **استخراج بيانات عبر جمل ال SQL.**
- **تغيير وضعية ال Data Files بين Offline و Online.**
- **القيام بعملية ال Recovery على ال Offline Data Files وال Tablespaces.**

Restricted Mode:

يمكن تشغيل ال Database في هذه الحالة لإعطاء المجال للمستخدمين الذي يملكون Restricted Session Privilege من القيام بمهات خاصة.

مثال تطبيقي 2.8:

لتشغيل ال Database في حالة ال Restricted :

```
STARTUP RESTRICTED;
```

يمكن تحويل الحالة بشكل ديناميكي باستخدام:

```
ALTER SYSTEM ENABLE RESTRICTED SESSION;
ALTER SYSTEM DISABLE RESTRICTED SESSION;
```

Kill a Session:

في بعض الأوقات (وخاصة بعد وضع ال Restricted) يريد ال DBA (أو من يملك الصلاحيات) باخراج بعض المستخدمين من ال Database للقيام ببعض المهمات. يمكن القيام بهذه المهمة عبر:

```
ALTER SYSTEM KILL SESSION 'SID, SERIAL#';
```

ال SID هو رقم ال Session الخاص (يعطى لكل Session عند بداية تشغيله رقم خاص به) يمكن استخراجه من V$SESSION.

ال SERIAL# هو رقم أخر مميز "Unique" يعطى لكل Session ويمكن استخراجه من ايضاً من ال V$SESSION.

مثال تطبيقي 2.9:

يمكن القيام بعملية "Kill Session" باتخاذ الخطوات التالية:

- يجب أن يكون هنالك أكثر من مستخدم يعملون على ال Database في نفس الوقت، للقيام بذلك ادخل الى SQLPLUS أولاً باسم مستخدم ثم افتح ال SQLPLUS مرة أخرى وادخل باسم مستخدم اخر.
استخدم SYSTEM و استخدم username = SCOTT , password =TIGER

- يتم كتابة الجملة التالية من ال SYSTEM:

```
SELECT USERNAME, SID, SERIAL#
FROM V$SESSION
WHERE USERNAME = 'SCOTT';
```

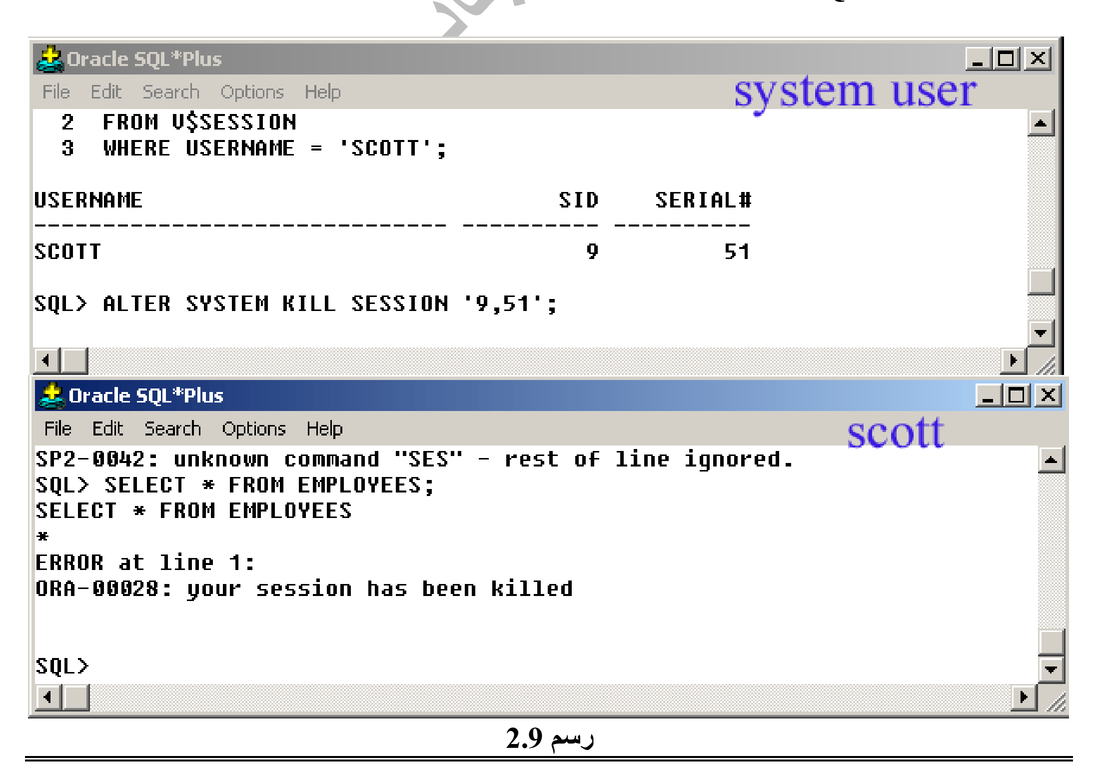
- يظهر الناتج التالي (يختلف من جهاز الى أخر)

USERNAME	SID	SERIAL#
SCOTT	9	51

- يكتب مستخدم ال SYSTEM التالي:

```
ALTER SYSTEM KILL SESSION '9, 51';
```

- يتم اغلاق ال Session الخاص بالمستخدم SCOTT. يقوم ال PMON بمهامه مباشرة لمعالجة الأثار التي يمكن ان تنجم عن أغلاق ال Session الخاص بالمستخدم SCOTT واذا قام SCOTT بطلب جملة SQL تظهر الرسالة التالية: YOUR SESSION HAS BEEN KILLED



رسم 2.9

SHUTTING DOWN THE DATABASE

يمكن اقفال ال Database في اربع حالات هي: NORMAL, TRANSACTIONAL, IMMEDIATE, ABORT.

ملاحظة: عادةً ما تكون عملية اغلاق ال Database من أجل عملية ال Backup.

ملاحظة2: لإغلاق ال Database تحتاج الى SYSDBA أو SYSOPER.

لإغلاق ال Database في احدى الحلات المذكورة تستعمل القاعدة التالية:

SHUTDOWN [NORMAL] | [TRANSACTIONAL] | [IMMEDIATE] | [ABORT];

مع العلم أن NORMAL هي ال Default، أي عند كتابة ال SHUTDOWN بدون أي من الحالات الأربع يفترض الأوراكل أنها الحالة ال Default.

:NORMAL

في حالة اغلاق ال Database بهذه الحالة يتم التالي:

- لا يسمح بدخول مستخدميين جدد لل Database.
- لا يغلق ال Database قبل خروج جميع المستخدمين منه، أي ان الأوراكل ينتظر حتى يفرغ المستخدميين من مهامهم ثم خروجهم لكي يتم اعلاق ال Database.
- يتم نقل بيانات ال Redo Log Buffer الى ال Redo Log Files.
- يتم نقل بيانات ال Database Buffer Cache الى ال Data Files.
- في حال وجود عمليات لم يتم حدوث COMMIT فيحدث لها ROLLBACK.
- انهاء وجود ال Background Processes.
- ازالة مساحة ال SGA من الذاكرة.
- اغلاق ال Database ثم اغلاق ال Instance.

ملاحظة: لا يستلزم ان تتم عملية Recovery للبيانات التي كانت موجودة في ال Instance باعتبار انها نقلت الى الملفات الفيزيائية. وعملية ال Recovery هي العملية المشابهه لوظائف الPMON و SMON .

:TRANSACTIONAL

في حالة اغلاق ال Database بهذه الحالة يتم جميع خطوات ال NORMAL عدا الإختلافات التالية:

- لا يغلق ال Database قبل انهاء المهمات التي طلبها جميع المستخدمين منه، أي أن الأوراكل ينتظر حتى يفرغ المستخدميين من مهامهم التي طلبوها قبل حدوث أمر الاغلاق ولا ينتظر أن يخرج المستخدمين من ال Database لتتم عملية الاغلاق كما يحدث في ال NORMAL.
- لا يسمح بالقيام بمهمات جديدة من قبل المستخدمين بعد أمر الإغلاق.
- يتم اخراج المستخدم من ال Database عند انتهاء المهمات التي طلبت قبل عملية الاغلاق.

ملاحظة: لا يستلزم ان تتم عملية Recovery عند تشغيل ال Instance مرة أخرى.

:IMMEDIATE

في حالة اغلاق ال Database بهذه الحالة يتم جميع خطوات ال NORMAL عدا الإختلافات التالية:

- يتم اغلاق ال Database فوراً دون انتظار خروج المستخدمين أو انهاء مهامهم التي طلبوها قبل أمر الاغلاق.
- المهمات التي تم طلبها قبل أمر الاغلاق تلغى وأية تعديلات طرأت على البيانات بسببها تزال وتمسح (بشرط ان تكون المهمة ما تزال تعمل عند أمر الإغلاق) ويحدث لها ROLLBACK.

ملاحظة: لا يستلزم ان تتم عملية Recovery عند تشغيل ال Instance مرة أخرى.

:ABORT

في بعض الحالات الخاصة عند عدم استطاعة الأوراكل اغلاق ال Database بأحد الحالات الثلاث السابقة يتم استخدام هذه الحالة.
في حالة اغلاق ال Database بهذه الحالة يتم التالي:

- يتم اغلاق ال Database فوراً دون انتظار خروج المستخدمين أو انهاء مهامهم التي طلبوها قبل أمر الاغلاق.
- المهمات التي تم طلبها قبل أمر الاغلاق تلغى و أية تعديلات طرأت على البيانات بسببها تزال وتمسح (بشرط ان تكون المهمة ما تزال تعمل عند أمر الإغلاق) ولا يحدث لها ROLLBACK.
- لا يتم نقل بيانات ال Redo Log Buffer الى ال Redo Log Files.
- لا يتم نقل بيانات ال Database Buffer Cache الى ال Data Files
- المهمات التي لم يحدث لها COMMIT لا يحدث لها ROLLBACK.
- يتم اغلاق ال Database وال Instance بشكل فوري.

ملاحظة: يستلزم ان تتم عملية Recovery عند تشغيل ال Instance مرة أخرى.

ملاحظة2: تأثيرات ال STARTUP FORCE و عند حدوث أي عطل في ال Instance يؤدي الى اغلاقه هي ذات التأثيرات لل SHUTDOWN ABORT.

DIAGNOSTIC FILES

تعتبر ملفات ال Diagnostic ملفات مهمة لإدارة ال Instance لأنها تخزن بيانات حول العمليات المختلفة التي تجرى على ال Database. يوجد عدد من ملفات ال Diagnostic منها:

- **ALERT LOG FILE: يتم تخزين البيانات حول العمليات أو المهام اليومية التي تجرى على ال Database.**
- **BACKGROUND TRACE FILES: يتم تخزين بيانات حول ال Background Processes.**
- **USER TRACE FILE: يتم تخزين بيانات حول الأخطاء الفادحة التي يقوم بها المستخدمون والتي قد تؤدي الى تعطل ال Database وبذلك تساعد على اكتشاف الاسباب التي أدت لتعطل ال Database.**

:ALERT LOG FILE

لكل Instance ملف Alert Log بحيث اذا لم يقم المستخدم بتكوينه يقوم الأوراكل خلال بدأ ال Instance بتكوين الملف (خلال مرحلة ال NOMOUNT) وهو يعرف باسم alertSID.log. يعتبر ال Alert Log هو المرجع الأول في البحث عن الأخطاء (تشخيص) التي حدثت في ال Database بشكل يومي. موقع ال Alert File محدد من قبل العامل "Parameter" BACKGROUND_DUMP_DEST.

تذكر: جاء ذكر العامل "Parameter" BACKGROUND_DUMP_DEST في موضوع ال SPFILE.

يتم تخزين البيانات عديدة في Alert Log File منها:

- **توقيت بدأ تشغيل ال Instance وتوقيت الاغلاق.**
- **توقيت بدأ تشغيل ال Background Processes.**
- **بيانات حول ال Log Switch.**
- **بيانات حول الاخطاء التي ظهرت "Error Messages".**
- **بيانات حول تكوين ال Tablespaces.**
- **بيانات حول تكوين ال Undo Segments.**
- **جمل ال SQL التي تحتوي على الأمر ALTER.**
- **رقم ال Log Sequence الخاص بال Online Redo Log File الذي يتم العمل عليه من قبل ال LGWR**

سوف يتم التطرق لاحقاً لل Undo Segments.

تذكر: جاء ذكر ال Log Sequence Number في الفصل الأول.

:BACKGROUND TRACE FILES

يتم تخزين بيانات حول الاخطاء التي تحدثها ال Background Processes. يتم تكوين الملفات فقط عندما تحدث الأخطاء (أي انه في حال عدم حدوث اخطاء لا تكون الملفات متواجده).
موقع الملفات محدد من قبل العامل "Parameter" BACKGROUND_DUMP_DEST.

:USER TRACE FILE

يتم تخزين بيانات حول الأخطاء الفادحة التي يقوم بها المستخدمون والتي قد تؤدي الى تعطل ال Database وبيانات حول ال SQL التي تم طلب مراقبتها "Trace". موقع الملف محدد من قبل العامل "Parameter" USER_DUMP_DEST.
يتحكم في حجم الملف العامل "Parameter" MAX_DUMP_FILE_SIZE و المحدد ب 10 ميغا بايت. يمكن تشغيل أو ايقاف الملف في ال Session الواحد او في ال Instance (كل ال Sessions التي موجودة في ال Instance) عبر كتابة التالي:

في ال Session:

```
ALTER SESSION SET SQL_TRACE = TRUE;
```

في ال Instance:

او عند وضع قيمة العامل "Parameter" التالي في ال Initialization Parameter File:

```
SQL_TRACE = TRUE
```

وبذلك يتم تعقب "Trace" جميع جمل ال SQL وتسجيل بيانات حولها والأخطاء التي سببتها.

تذكر: جاء ذكر العامل "Parameter" USER_DUMP_DEST في موضوع ال SPFILE.

تعرف ملفات ال Trace Files بالرمز (trc). فمثلاً ملفات ال Background Trace تسمى بالقاعدة التالية: (كما الملفات الموجودة في نسخة الأوراكل الخاصة بي) sidPROCESSNAME.trc مثل dbnamePMON.trc (dbname هو اسم ال Instance في جهازي والذي يحدد عند عملية التنصيب).

وكذلك الحال بالنسبة لملفات ال User Trace في تسمى بالقاعدة التالية في نسخة الأوراكل الخصة بي: ORAPID.trc مثل ORA00853.trc.

ملاحظة: قد تختلف قاعدة التسمية من نسخة أوراكل الى أخرى ولكن ال Trace Files تنتهي دائماً ب (trc.)

ORACLE MANAGED FILES (OMF)

تسهل ال OMF ادارة عدد من ملفات الأوراكل اذ يكفي تحديد مواقع الملفات فيقوم الأوراكل بالاهتمام بالملفات و تكوينها وحذفها وتسميتها. يمكن تحديد مواقع الملفات باستخدام عاملين "Parameters" اثنين جديدن من عوامل ملف ال Initialization Parameter File اضيفا في النسخة التاسعة من الأوراكل "Oracle9i" هما:

- DB_CREATE_FILE_DEST: الذي يحدد مواقع ال Data Files.
- DB_CREATE_ONLINE_LOG_DEST_n: يحدد مواقع ملفات ال Redo Log وال Control Files. يدل الحرف (n) على رقم العامل "Parameter" بحيث يكون من واحد الى خمسة.

ملاحظة: يمكن استخدام أحدهما أو كلاهما, ولا يشترط استخدام الاثنين معاً.

ترتبط كل Tablespace بملف من ملفات ال Data Files (Physical-Logical) بحيث يكون لكل Tablespace ارتباط بملفات من Data Files. اي انه عند حذف ال Tablespace يجب حذف ملف ال Data Files المرتبط معها وإلا قد يسبب مشاكل عند عملية ال Backup، اذ انه لا فائدة من حفظ ال Data Files باعتبار ال Tablespace تم حذفها.
يجب على ال DBA البحث عن الملف المرتبط وحذفه بطريقة عادية (كما يتم حذف أي ملف في الويندوز أو ال UNIX أو غيرها). أما اذا تم استخدام طريقة ال OMF يقوم الأوراكل بحذف ال Data Files دون تدخل ال DBA أي بطريقة أوتوماتيكية وهذه أحد فوائد استخدام ال OMF.

يتم استخدام (_ora) في بداية اسماء ملفات ال OMF للدلالة عليها.

مثال تطبيقي 2.10:

```
DB_CREATE_FILE_DEST = 'C:\oracle\ora92\oradata\u01'

DB_CREAE_ONLINE_LOG_DEST_1 = 'C:\oracle\ora92\oradata\u02'

DB_CREAE_ONLINE_LOG_DEST_2 = 'C:\oracle\ora92\oradata\u03'
```

يمكن تغيير القيم "Values" بشكل دايناميكي باستخدام ALTER SYSTEM.

```
ALTER SYSTEM SET
DB_CREATE_FILE_DEST='C:\oracle\ora92\oradata\u04';
```

بعد ان يتم تحديد العوامل "Parameter" (كما في المثال 2.10) تتم عملية تكوين ال Database باستخدام أمر CREATE DATABASE. اذا فشل تكوين ال Database يقوم الأوراكل بحذف الملفات التي تكونت من خلال OMF.

يمكن الحصول على معلومات حول الملفات التي تكونت بطريقة ال OMF من DBA_DATAFILE و V$LOGFILE.

الفصل الثالث

تكوين الداتا بيس واستخدام الداتا ديكشينوري

CREATING A DATABASE & USING THE DATA DICTIONARY

PRE-CREATING STAGES

عملية تكوين ال Database تحتاج الى تخطيط وتحضير مسبق، يجب على ال DBA اعداد التالي:

- الحجم الكافي لكل من القرص الصلب "Hard Disk" و الذاكرة في السيرفر.
- دراسة نوعية النظام "Operating System" الموجود في السيرفر.
- وضع خطة لتوزيع الملفات في موقع التخزين لحماية ملفات ال Database.
- تحديد العوامل Initialization Parameters وخاصة ال DB_BLOCK_SIZE الذي لا يمكن تغييره بعد عملية التنصيب.
- اختيار نظام الحماية (Operating System او Password File) و توفر ال SYSDBA.
- معرفة اذا كان يراد تكوين Database جديدة أو يراد نقل ال Database الموجودة من نسخة أوراكل أقدم الى نسخة أجد ، عند ذلك يجب استخدام ال Data Migration Assistant
- تحديد الطريقة التي سوف يتم بها تكوين ال Database الجديدة ، إما عن طريق استخدام CREATE DATABASE في ال SQL (تسمى الطريقة اليدوية) او عن طريق Database Configuration Assistant.
- وضع خطة عملية دورية لإجراء عملية ال Backup.

نقاط مهمة:

1- ينصح بتوزيع ملفات ال Control File على مواقع مختلفة أو أقراص صلبة مختلفة.

2- ينصح بتوزيع الملفات المتشابة في مجموعات ال Redo Log الى مواقع مختلفة أو أقراص مختلفة.

3- وضع نظام لتسمية الملفات ليتسنى ايجادها بسهولة.

4- تفريق عناصر ال Database التي لها مراحل حياة مختلفة(قصيرة أو طويلة) الى مواقع مختلفة لكي يقل ما يعرف باسم Disk Fragmentation.

5- تفريق ال Tables وال Indexes الى Tablespaces مختلفة لتنظيم عملية ال Input/Output.

DATABASE CONFIGURATION ASSISTANT

يمكن عبر ال(DBCA) القيام بالتالي: Create a Database, Configure Database Options, Delete a Database, and Manage Templates.

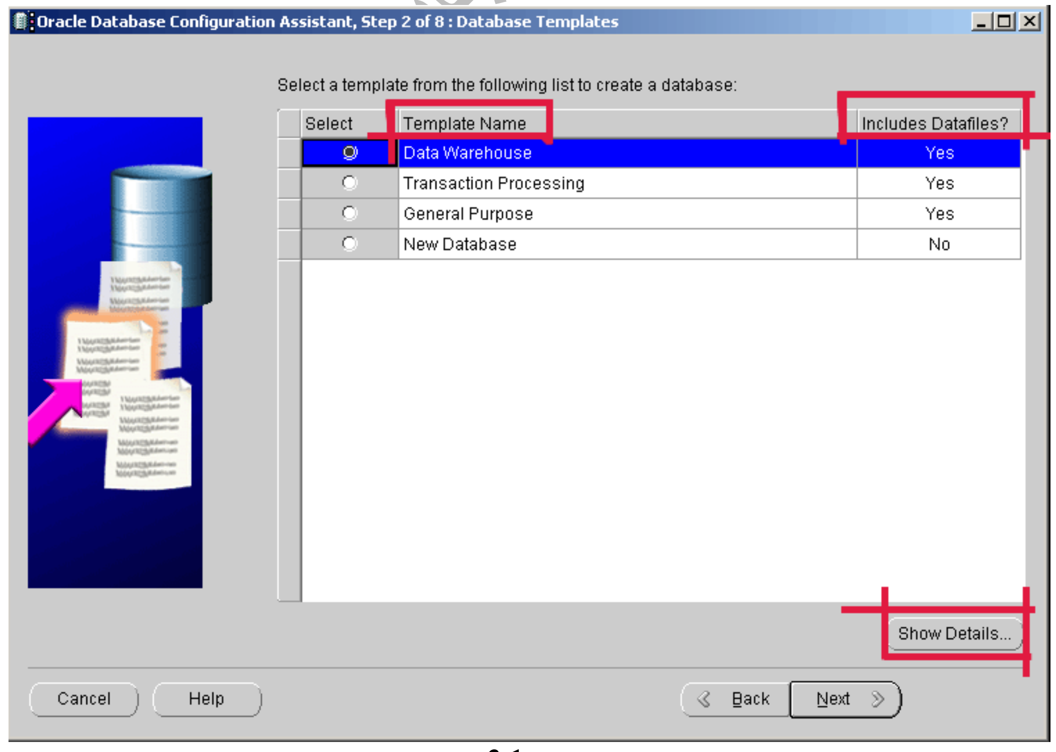
[CREATE A DATABASE:]

للقيام بتكوين Database جديدة أو القالب "Template". والقالب عبارة عن اختيار ال Database Options وحفظها لكي يتسنى استخدامها في موقع أخر. مثلاً يمكن ان يقوم الفرع الأساسي للشركة بتكوين قالب ال Database وتوزيعه على الفروع لتكوين Databases متشابه في جميع الفروع.
يمكن تكوين نوعين من القوالب "Templates" هما:

- **بدون ال Data Files: فقط اختيارات ال Database دون الملفات (لمعرفة الاخيارات اضغط على "Show Details"). تكمن فائدته انه يمكن تغيير جميع الأختيارات وال Initialization Parameters بعد تكوين القالب "Template".**

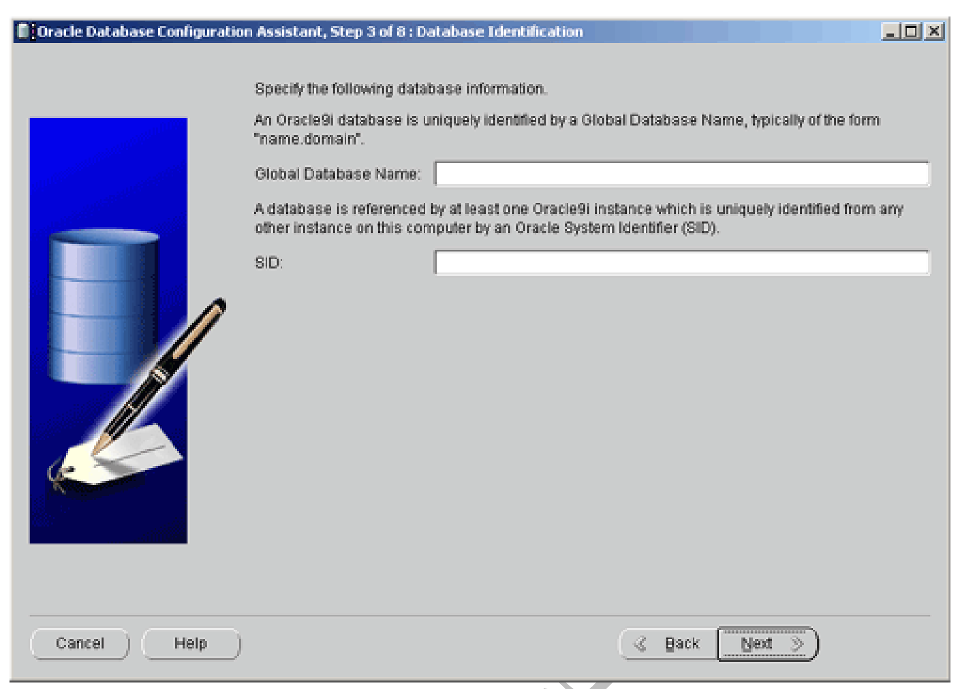
- **مع ال Data Files: يكون القالب متكوناً من اختيارات ال Database مع وجود ملفات ال Data Files. يتم تكوين ال Control Files و ال Redo Log Groups بشكل أوتوماتيكي، يمكن تغيير اسماء ومواقع ال Data Files، ولكن لا يمكن حذف ال Data Files وتكوينها من جديد وكذلك ينطبق الحال على Tablespaces، ولا يمكن تعدل ال Initialization Parameters بعد تكوين القالب "Template".**

يوجد عدد من القوالب المعدة مسبقاً "Template Name" تظهر بالرسم 3.1:



رسم 3.1

بعد أن يتم اختيار أحد القوالب "Templates" يطلب اختيار اسم ال Database واسم ال Instance المعرف بالرمز SID.



رسم 3.2

في حال تم اختيار القالب "New Database" يتم اختيار المميزات "Features" التي تريد استخدامها في ال Database.

Oracle Database Configuration Assistant, Step 4 of 8 : Database Options

Database Features | Custom Scripts

Select the features you want to configure for use in your database:

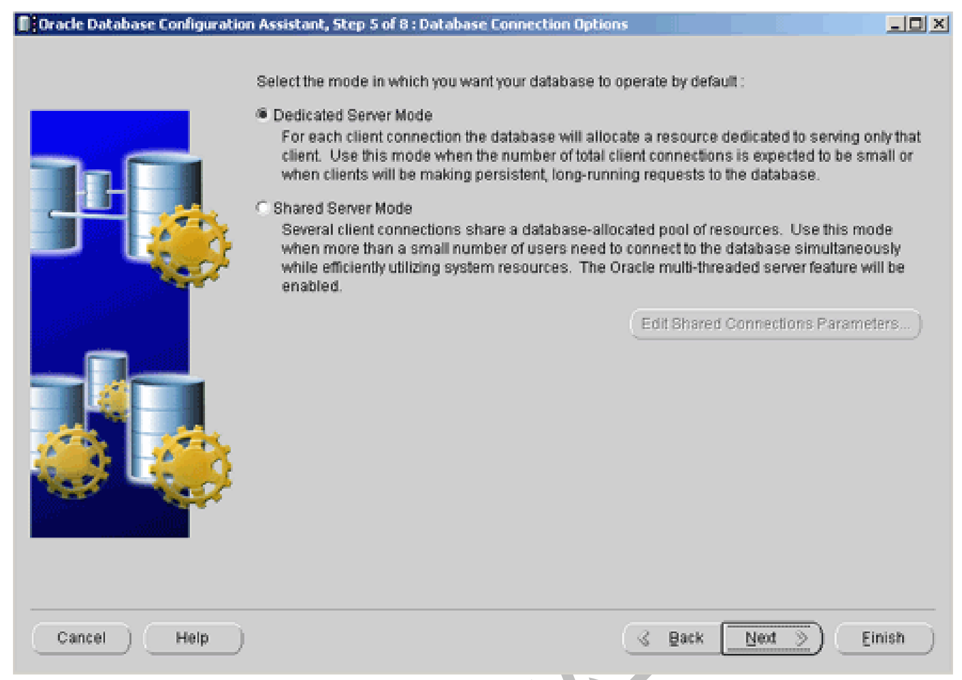
- ☑ Oracle Spatial
- ☑ Oracle Ultra Search
- ☐ Oracle Label Security
- ☑ Oracle OLAP Services
- ☑ Example Schemas
  - ☑ Human Resources
  - ☑ Order Entry
  - ☑ Product Media
  - ☑ Sales History
  - ☑ Shipping

Additional database configurations...

Cancel | Help | Back | Next | Finish

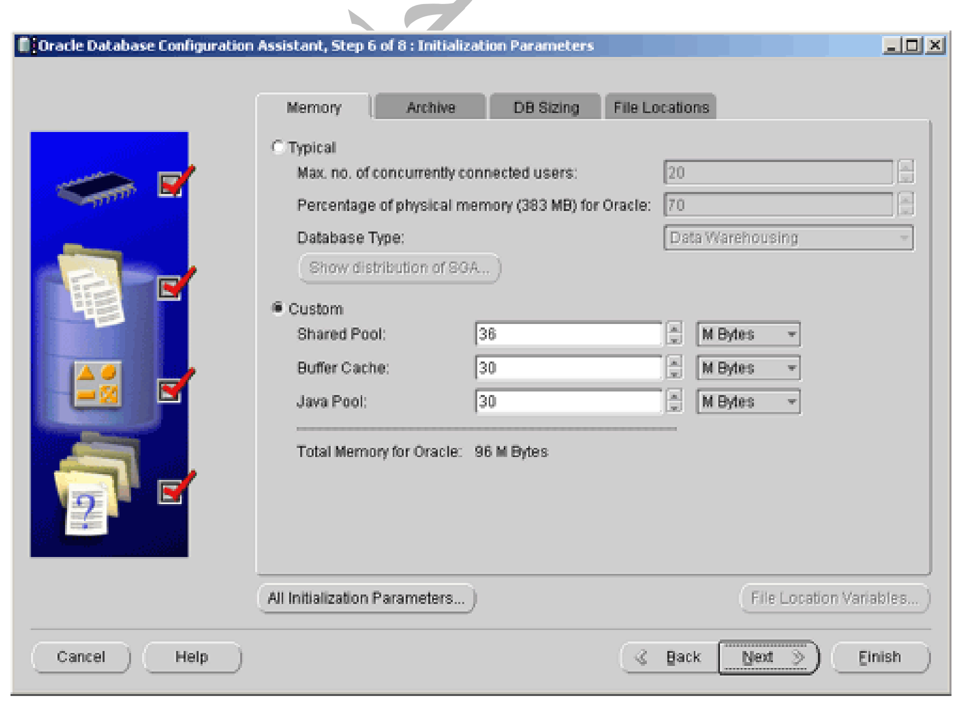
رسم 3.3

المرحلة التالية يتم اختيار البيئة التي سوف يعمل بها ال Database.



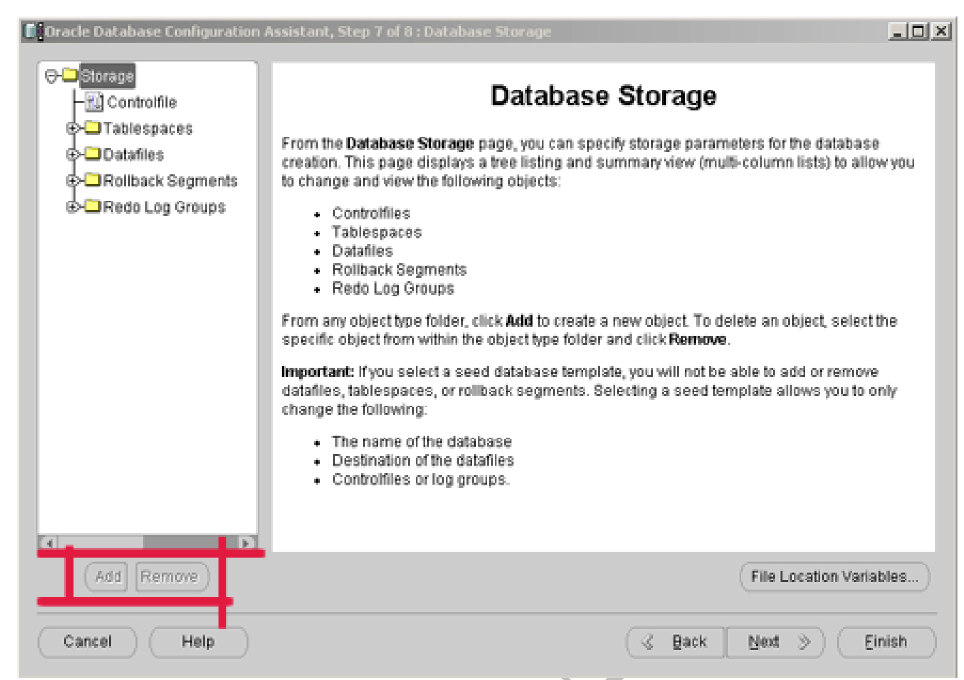
رسم 3.4

بعد ذلك يتم تحديد عوامل "Parameters" ال Initialization Parameter File المختلفة.



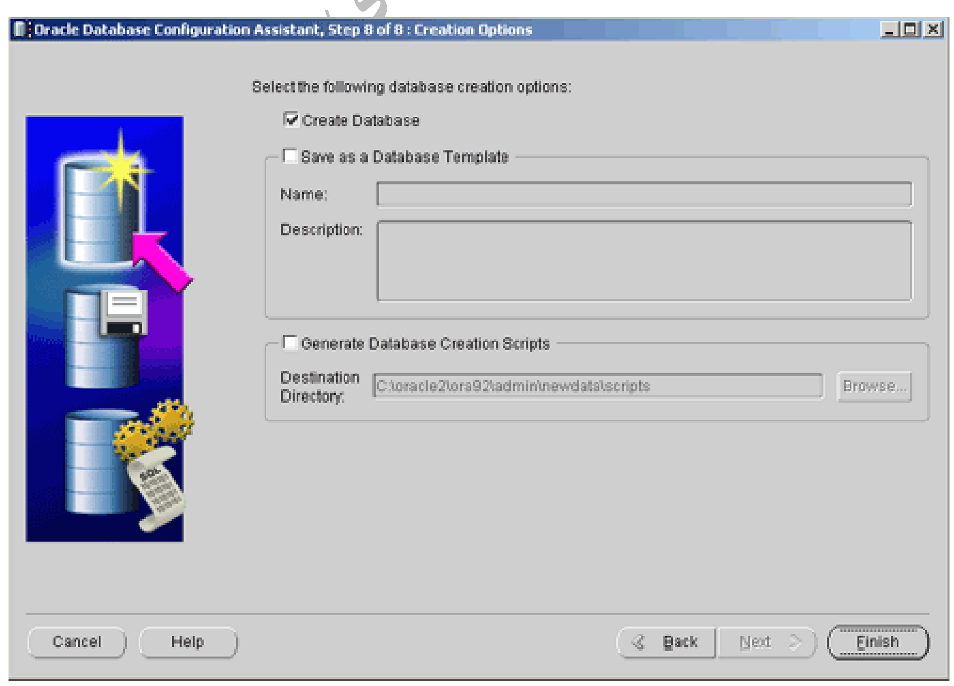
رسم 3.5

يتم بعد ذلك عرض اسماء الملفات الفيزيائية وال **Tablespaces** ومواقعها ولديك الخيار بحذف أو اضافة ملفات اخرى.



رسم 3.6

في المرحلة التالية يتم اختيار نوعية التكوين، هل تريد تكوين **Database** أو تريد تكوين فقط قالب **"Template"** أو هل تريد تكوين جمل **SQL Scripts** التي يمكن استخدامها مع أمر **CREATE DATABASE**.



رسم 3.7

:CONFIGURE DATABASE OPTIONS

يمكن تعديل ال Database التي تم تكوينها سابقاً من ناحية ال Oracle Features والبيئة التي تعمل عليها (Dedicated Server أو Shared Server).

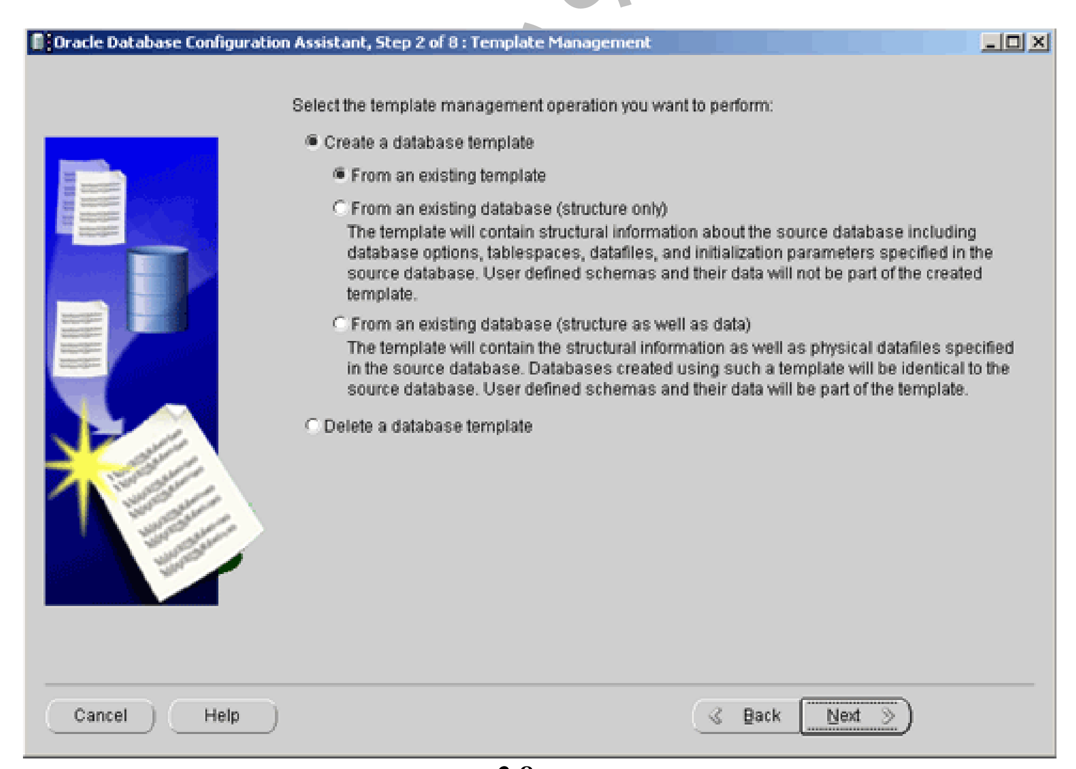
:DELETE A DATABASE

لحذف Database تم تكوينها سابقاً

:MANAGE A TEMPLATE

يمكن استخدامها لتكوين قالب "Template" أو لحذف قالب "Template" موجود. عند تكوين قالب "Template" جديد يمكن الاختيار بين ثلاث خيارات هي:

- من قالب "Template" أخر موجود، اذ يمكن تعديل مواصفات القالب "Template" لصنع قالب "Template" جديد.
- من مواصفات Database موجودة (تم تكوينها سابقاً) ، اذ يمكن تكوين قالب "Template" مشابه لمواصفات ال Database ولكن دون ملفات ال Data Files.
- من Database موجودة، اذ يمكن تكوين نسخة مطابقة ل Database موجودة مع ال Data Files وجميع البيانات.



رسم 3.8

CREATE A DATABASE MANUALLY

يمكن تكوين ال Database بطريقة يدوية باستخدام الأمر CREATE DATABASE.
الخطوات المتبعة تتلخص بالتالي:

- **تحديد اسم ال Instance وال Database.**
- **تحديد ال Database Character Set التي سوف نتطرق لها لاحقاً في فصل أخر.**
- **تجهيز ملف ال Initialization Parameter File.**
- **تشغيل ال Database في الوضعية NOMOUNT، عبر ال SYSDBA.**
- **كتابة الأمر CREATE DATABASE.**
- **تشغيل ال SQL SCRIPTS لتكوين ال Data Dictionary .**

ملاحظة: اذا كنت تقوم بتكوين ال Database على نظام ال UNIX يجد تحديد ما يعرف باسم ال Environmental Variables ومنها :

ORACLE_BASE: هو الموقع "Directory" الذي يحوي كل ملفات الأوراكل "Top of Tree".
وهنا يتم تخزين كل نسخ الأوراكل المختلفة "Versions".

ORACLE_HOME: موقع ال Oracle Software.

ORACLE_SID: تحدد اسم ال Instance والذي يجب ان يكون غير متشابه مع اسم Instance أخر.

ORA_NLS33: لتحديد "Database Character Set" غير التي تم تحديدها مسبقاً "Default".

PATH: مواقع برامج الأوراكل مثل ال SQLPLUS، ومن المفترض ان تكون $ORACLE_HOME\bin.

LD_LIBRARY_PATH: ملفات أخرى تسمى Oracle Library Files وعادة ما تكون $ORACLE_HOME\lib.

يمكن تكوين Database بشكل يدوى باستخدام القاعدة التالية:

```
CREATE DATABASE [name of database]
[CONTROLFILE REUSE]
[LOGFILE [GROUP number] C1
[MAXLOGFILES number]
[MAXLOGMEMBERS number]
[MAXLOGHISTORY number]
[MAXDATAFILES number]
[MAXINSTANCES number]
[ARCHIVELOG | NOARCHIVELOG]
[CHARACTER SET char]
[NATIONAL CHARACTER SET char]
[DATAFILE C1 [C2]]
[DEFAULT TEMPORARY TABLESPACE tablespace_name C1 [C3]]
[UNDO TABLESPACE tablespace_name DATAFILE C1 [C2]]
[SET TIME_ZONE [time_zone_region]]
```

أما ال الرموز ال C1، C2،C3 فترمز الى جزء من القاعدة تم تخفيفة الى رموز لسهولة قراءة القاعدة:

C1= 'filename' [SIZE number] [K | M] [REUSE]

C2= [AUTOEXTEND OFF | ON [NEXT number [K | M]]
[MAXSIZE UNLIMITED | number [K | M]]]

C3= EXTENT MANAGEMENT LOCAL UNIFORM [SIZE number] [K | M]

اي انه في الجملة الثالثة تكون القاعدة:

[LOGFILE [GROUP number] 'filename' [SIZE number] [K | M] [REUSE]

ملاحظة: جملة ال CREATE DATABASE هي الجملة الإلزامية الوحيدة و الباقي اختياري.

ملاحظة2: عند عدم استخدام كلمة ال SIZE يجب استخدام كلمة ال REUSE والعكس صحيح.

أما أوامر القاعدة فهي :

- [name of database]: اسم ال Database المراد تكوينه. في حال عدم كتابته يؤخذ قيمة العامل DB_NAME.
- [CONTROLFILE REUSE]: يمكن استخدامها لطلب اعادة استخدام "Overwrite" ملفات ال Control Files المعرفة في ال Initialization Parameter File عوضاً عن تكوين ملفات جديدة.
- [MAXLOGFILES number]: تحدد العدد الأقصى لمجموعات ال Redo Log Files التي يمكن تكوينها في ال Database على مدى حياة ال Database، العدد الأدنى هو 2.
- [MAXLOGMEMBERS number]: تحدد العدد الأقصى لملفات ال Redo Log ضمن المجموعات.
- [MAXLOGHISTORY number]: تحدد عدد ال Archived Log Files التي تسخدم في عملية ال Media Recovery.
- [AUTOEXTEND]: سوف نتطرق لها لاحقاً.
- [MAXDATAFILES]: تحدد العدد الأقصى من ال Data Files الذي يمكن تكوينه على مدى حياة ال Database.
- [MAXINSTANCES]: تحدد العدد الأقصى من ال Instances الذين يستطيعون العمل على ال Database في نفس الوقت.
- [CHARACTER SET char]: سوف نتطرق لها لاحقاً.
- [NATIONAL CHARACTER SET]: سوف نتطرق لها لاحقاً.
- [DEFAULT TEMPORARY TABLESPACE]: لتكوين ال Default Temporary Tablespace التي سوف نتطرق لها لاحقاً، اذا لم تحدد يقوم الأوراكل بتكوينها للمستخدم.
- [UNDO TABLESPACE]: لتكوين ال Undo Tablespace التي سوف نتطرق لها لاحقاً.
- [SET TIME_ZONE]: تحديد وقت ال Database.

مثال تطبيقي 3.1:

```
CREATE DATABASE db01
CONTROLFILE REUSE
LOGFILE
GROUP 1 ('C: \oracle\ora92\oradata\db01\log101.log') SIZE 10M
GROUP 2 ('C: \ oracle\ora92\oradata\db01\log201.log') SIZE 10M
GROUP 3 ('C: \ oracle\ora92\oradata\db01\log301.log') SIZE 10M
MAXLOGFILES 4
MAXLOGMEMBERS 2
MAXLOGHISTORY 0
MAXDATAFILES 254
MAXINSTANCES 2
ARCHIVELOG
CHARACTER SET AL32UTF8
NATIONAL CHARACTER SET AL16UTF16
DATAFILE 'C: \oracle\ora92\oradata\db01\system01.dbf' SIZE 100M
AUTOEXTEND ON NEXT 5M MAXSIZE UNLIMITED
UNDO TABLESPACE UNDO01
DATAFILE 'C: \oracle\ora92\oradata\db01\undo01.dbf' SIZE 40M
DEFAULT TEMPORARY TABLESPACE TEMP01 TEMPFILE
'C: \oracle\ora92\oradata\db01\temp01.dbf' SIZE 20M
EXTENT MANAGEMENT LOCAL UNIFORM SIZE 128K
SET TIME_ZONE = 'CANADA /YUKON'
;
```

الأخطاء التي قد تؤدي الى فشل تكوين ال Database بشكل يدوي:

- أخطاء في كتابة جمل ال SQL.
- طلب تكوين ملفات موجودة مسبقاً (تأكد من ان اسم الملف غير مستخدم)
- مشاكل أو اخطاء ناتجة من نظام التشغيل "Operating System".

لأعادة تكوين ال Database بعد محاولة فاشلة يجب:

- اغلاق ال Instance.
- حذف أي ملفات تكونت من ال CREATE DATABASE التي فشلت.
- تصحيح سبب الفشل ، مثل تعديل جملة ال SQL التي تحوي أخطاء.
- تشغيل ال Instance في وضعية ال NOMOUNT والقيام تكوين ال Database من جديد.

ملاحظة: لا يتكون ال Data Dictionary بعد الانتهاء من تكوين ال Database بشكل يدوي، ولكن يتم تكوين ال Dynamic Views مثل ال V$DATAFILE.

تذكر: تغيير كلمة السر لكل من المستخدمين SYS و SYSTEM لأخذ الأمان.

تذكر: ان استخدام ال Oracle Managed Files (OMF) يسهل عملية تكوين ال Database.

DATA DICTIONARY

يعتبر ال Data Dictionary أحد أهم مكونات ال Database، اذ يتكون من مجموعة من ال Tables وال Views التي تحتوي معلومات قيمة عن ال Database. تستخدم ال Data Dictionary لقراءة البيانات فقط ولا يمكن تعديل بياناتها بواسطة ال DBA، ولكن يقوم ال Oracle بتعديل البيانات الموجودة عند حدوث أوامر ال Data Definition Language (DDL). يستخدم الأوراكل ال Data Dictionary ايضاً عند طلب المستخدم الدخول الى ال Database ، اذ يقوم بالتأكد من وجود اسم المستخدم وبيانات المستخدم مثل ال User Privileges.

يتكون ال Data Dictionary من عنصرين هما:

- **BASE TABLES: يتم تخزين بيانات حول ال Database في ال Base Tables. تعتبر <u>أول</u> جزء "Object" من ال Database يتم تكوينه، ويتم تكوينها بشكل أتوماتيكي من قبل الأوراكل باستخدام الملف "Script" المسمى باسم sql.bsq. لا يجب تعديل ال Base Tables أوالعبث بها لأنها ضرورية لتشغيل ال Database، ولكن يمكن تعديل واحدة فقط وهي ال AUD$ ، مع العلم أنه يتم تخزين البيانات في ال Base Tables <u>بشكل مشفر</u>.**

- **DICTIONARY VIEWS: تقوم بتلخيص وترتيب بيانات المخزنة في ال Base Tables لكي تسهل استخراج البيانات و قرائتها. يتم تكوينها باستخدام الملف "Script" المسمى باسم catalog.sql.**

<u>بيانات ال Data Dictionary:</u>

يوفر ال Data Dictionary بيانات مهمة حول:

- **معلومات تفصيلية حول جميع عناصر "Objects" ال Database مثل ال Tables, Indexes, Views, Synonyms, Data files, Control Files وغيرها.**
- **بيانات حول المساحة المشغولة والمساحة الفارغة في ال Database.**
- **بيانات حول المستخدمين "Users".**
- **بيانات تفصيلية لل Privileges و ال Roles الممنوحة لكل مستخدم.**
- **بيانات حول ال Auditing التي سوف نتطرق لها في فصل أخر.**
- **بيانات حول ال Integrity Constraint .**

QUERYING DATA DICTIONARY

تتوزع ال Dictionary Views الى ثلاث مجموعات عي:

- **DBA_: تحتوي ال Views على جميع بيانات العناصر "Objects" في ال Database، وهي خاصة لل DBA و كل مستخدم لديه ال Privileges اللازمة للدخول الى بياناتها.**

- **ALL_: تحتوي على بيانات حول جميع العناصر "Objects" التي يستطيع المستخدم استخراج البيانات منها، سواء ما كان منها ضمن ال Schema الخاصة بالمستخدم أو مستخدمين أخرين أو ال Public Schema.**

- **USER_: تحتوي على بيانات حول العناصر "Object" التي يملكها المستخدم (أي العناصر في ال Schema الخاصة بالمستخدم).**

يمكن معرفة جميع ال Views المتوفرة في ال Data Dictionary من خلال البيانات المعروضة من ال DICTIONARY view.

مثال تطبيقي 3.2:

```
SELECT * FROM DBA_OBJECTS;

SELECT * FROM ALL_OBJECTS;

SELECT * FROM USER_OBJECTS;

SELECT * FROM DICTIONARY;
```

DYNAMIC PERFORMANCE

طوال فترة عمل ال Database، يقوم Oracle Server بتسجيل بيانات حول عمل ال Database في الذاكرة على شكل Tables وعند اغلاق ال Database أو اغلاق السيرفر تضيع البيانات من الذاكرة.
تعتبر ال Dynamic Performance View من ضمن ال SYS Schema، وهي تستخدم بشكل اساسي لمراقبة ال Database ولا يسمح القيام بأوامر Data Manipulation Language(DML) عليها.

لمعرفة ال Views المتوفرة يمكن استخدام V$FIXED_TABLE، ولمعرفة ال Columns ضمن ال Views استخدم V$INSTANCE.

ملاحظة: يرمز لل Dynamic Performance Views بالرمز V$.

الفصل الرابع

ملفات الكنترول و الريدو لوج

CONTROL & REDO LOG FILES

CONTROL FILE

يعتبر Control File ضروري جداً لتشغيل وعمل ال Database، حيث تتم قرائته من قبل ال Oracle Server قبل بدء تشغيل ال Database، اذ يحتوي على بيانات مهمة حول ال Database مثل اسماء ومواقع ملفات ال Data Files. تتم تغير بياناته أو تجديدها "Update" عبر ال Oracle Server فقط ، اذ لا يستطيع أي مستخدم أو ال DBA القيام بهذه المهمة.
يتم تكوين ال Control File بعد تكوين ال Database مباشرة ولا يستطيع ال Control File الواحد العمل على أكثر من Database واحدة ، ويلزم Control File واحد فقط على الأقل لتشغيل ال Database

ملاحظة: تسمى عملية تكوين أكثر من نسخة متطابقة من ملفات ال Control Files (ينطبق الحال على ال Redo Log ايضا) بعملية ال Multiplex ، ويفضل توزيعها على مواقع أو اقراص صلبة مختلفة.

ملاحظة2: ملفات ال Control Files من نوعية ال Binary.

تذكر: تتم قراءة ملفات ال Control File في مرحلة ال Mount.

تذكر2: ملف ال SPFILE من نوعية ال Binary.

يوجد عوامل "Parameters" تحدد حجم ال Control File وهي:

- MAXLOGFILES
- MAXLOGMEMBERS
- MAXLOGHISTORY
- MAXINSTANCES
- MAXDATAFILES

يحتوى ال Control File على البيانات التالية:

- اسم ال Database التي يعمل لها ال Control File.
- تاريخ وتوقيت تكوين ال Database.
- اسماء ومواقع ملفات ال Data Files و ال Redo Log.
- اسماء ال Tablespaces.
- معلومات حول ال Checkpoint.
- معلومات حول Archived Log File.
- معلومات حول عملية ال Backup.
- توقيت بدء و توقف ال Undo Segment.
- الرقم الحالي ل Log Sequence Number والذي يتم تخزينه عند حدوث Log Switch.

تذكر: يحتوي ال alertSID.log ايضاً على Log Sequence Number.

MULTIPLEXING CONTROL FILES

للحماية من أية خلل قد يحدث لل Database نتيجة تعطل ال Control File ينصح بشدة بالقيام بعملية ال Multiplex لل Control File بيحث يتم تخزين النسخ المتطابقة في مواقع تخزين مختلفة لكي يتسنى استعادة Control File في حال تعطله أو ضياع بياناته.
للقيام بعملية ال Multiples يجب استخدام إما ال SPFILE أو ال init.ora .

ملاحظة: يمكن عمل ثمانية نسخ لل Control File.

:MULTIPLEXING USING SPFILE

للقيام ب Control File Multiples بواسطة ال SPFILE يجب إتباع الخطوات التالية:

- تغيير ملف ال SPFILE كما في المثل التالي:

```
ALTER SYSTEM SET CONTROL_FILES =
 ('...\ora01\oradata\db01\ctr001.ctl', '...\ora02\oradata\db01\ctr002.ctl',
'...\ora03\oradata\db01\ctr003.ctl', '...\ora04\oradata\db01\ctr004.ctl')
SCOPE=SPFILE;
```

- لكي تتم التغيرات يجب اغلاق ال Database بكتابة التالي:

```
SHUTDOWN;
```

- نسخ ملف ال Control File الى المواقع التي حددت بالنقطة الأولى (القيام بعملية النسخ كما في نظام التشغيل ، اي في الويندوز يستخدم الفأرة مع COPY و PASTE).

- تشغيل ال Database بكتابة التالي:

```
STARTUP
```

:MULTIPLEXING USING init.ora

للقيام ب Control File Multiples بواسطة ال init.ora يجب إتباع الخطوات التالية:

- يجب اغلاق ال Database بكتابة التالي:

```
SHUTDOWN;
```

- نسخ ملف ال Control File الى المواقع الجديدة التي سوف تحدد في النقطة الثالثة.

- تعديل العامل "Parameter" CONTROL_FILES في ملف ال init.ora كما في المثال:

```
CONTROL_FILES =(...\disk1\ctr01.ctl, ...\disk2\ctr02.ctl)
```

- تشغيل ال Database.

ملاحظة: استخدام الثلاث نقاط (...) في المثال السابق للاختصار فقط.

USING OMF:

يمكن تسهيل إدارة ملفات ال Control File باستخدام ال Oracle Managed Files. يمكن تكوين ال Control Files طبقاً لطريقة ال OMF بشكل أتوماتيكي عند تكوين ال Database اذا لم يتم تحديد العامل "Parameter" CONTROL_FILES في ال Initialization Parameter Files وتم تحديد عامل "Parameters" ال OMF DB_CREATE_ONLINE_LOG_DEST_n.
بعد تكوين ال Control File بطريقة ال OMF يجب تسجيل اسم وموقع ال Control File في ال PFILE ، اما اذا كنت تستخدم ال SPFILE فإن الاسماء تسجل بشكل اتوماتيكي.

DATA DICTIONARY & CONTROL FILE:

يمكن استخراج بيانات ال Control File من ال Data Dictionary عبر:

- V$CONTROLFILE: تعرض اسماء ال Control Files والحالة الحالية (STATUS) اذ انها دائما فارغة إلا في حال ضياع ملف من ملفات ال Control File فتصبح INVALID.

```
SELECT STATUS, NAME FROM V$CONTROLFILE;
```

- V$PARAMETER: تعرض اسماء وقيم "Values" جميع العوامل "Parameter" من ضمنها طبعاً ال Control Files.

```
SELECT NAME, VALUE FROM V$PARAMETER
WHERE NAME = 'control_files';
```

- V$CONTROLFILE_RECORD_SECTION: تعرض بيانات عن الأجزاء المختلفة من ال Control File.

```
SELECT TYPE, RECORD_SIZE, RECORDS_TOTAL, RECORD_USED
FROM V$CONTROLFILE_RECORD_SECTION;
```

- SHOW PARAMTER: يمكن استخدام هذه للحصول على مواقع واسماء ال Control Files

```
SHOW PARAMETER CONTROL_FILES;
```

ملاحظة: يمكن الحصول ايضاً على اسماء ال Control Files بعد تكوينها من ملف ال alertSID.log.

REDO LOG FILES

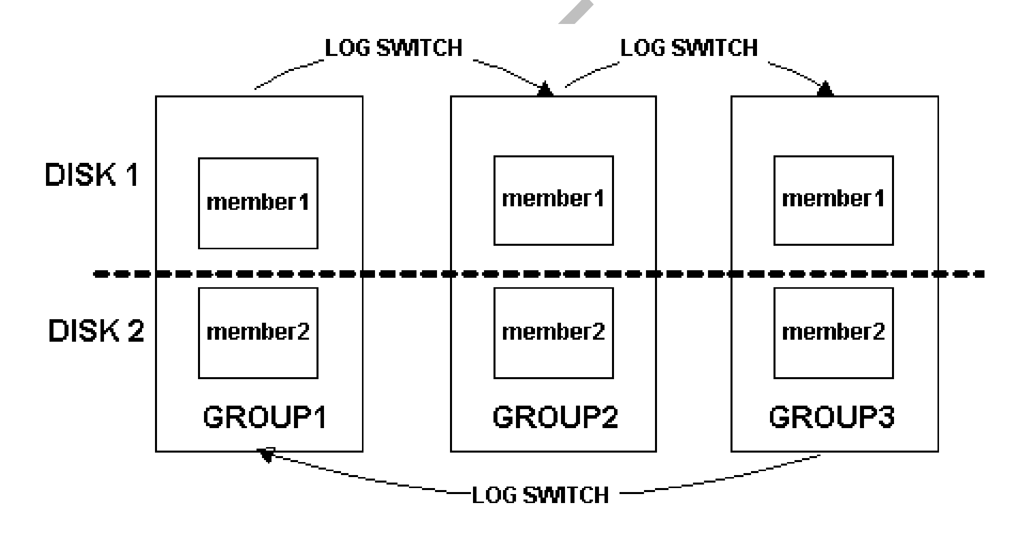
تعتبر ملفات ال Redo Log ضرورية جداً لعمل ال Database ، اذ يتم تخزين كل ما جرى ويجري على البيانات من تغيرات لكي يتم استعادة البيانات "Recovery" في حالة حدوث ضياع للبيانات عند حدوث عطل مفاجئ. تنقسم ملفات ال Redo Log الى مجموعات "Groups" كل مجموعة تحوي أعضاء "Members"، و الأعضاء "Members" ضمن المجموعة الواحدة عبارة عن نسخ متطابقة "Copies" تحوي على نفس البيانات ويتم نقل البيانات لها في نفس الوقت.

يتطلب الأوراكل على الأقل مجموعتين من مجموعات ال Redo Log Files وكل عضو "Member" ضمن المجموعة يكون له نفس الحجم وذات ال Log Sequence Number.

يقوم الأوراكل بواسطة ال LGWR بنقل بيانات ال Redo Log Buffer الى ال Redo Log Files حيث تتم الكتابة على مجموعة واحدة فقط وعندما تمتلئ المجموعة الأولى يقوم ال LGWR بكتابة البيانات في المجموعة الثانية وبعد إمتلاء المجموعة الأخيرة يعود ال LGWR بالكتابة في المجموعة الأولى ، وتسمى هذه العملية بال Log Switch وتسمى المجموعة التي يتم نقل البيانات لها باسم المجموعة الحالية أو "Current Online Redo Log Group".

ملاحظة: يمكن تحديد الحد الأعلى للمجموعات و الأعضاء بواسطة MAXLOGMEMBERS و MAXLOGFILES في جملة ال CREATE DATABASE.

تذكر: يتم تخزين بيانات حول ال Log Switch في ال Alert Log File.



رسم 4.1

كل مجموعة Redo Log File معرفة برقم مميز يسمى Log Sequence Number يمنحه الأوراكل لكل مجموعة عند بداية كتابة البيانات بها حيث يتم تجديد الرقم "Overwritten" كل مرة يتم إعادة استخدام المجموعة.

تذكر: يتم تخزين Log Sequence Number في:

- Control File
- Data Files Header

- Alert Log File

:Checkpoint

تعرضنا لل Checkpoint في الفصل الأول وذكرنا انه يقوم بكتابة ال Log Sequence Number في ال Control File. ونتطرق لل Checkpoint مرة أخرى لأن العلاقة بين ال Checkpoint وال Log Switch متزامنة بحيث عند حدوث ال Log Switch تحدث عملية ال Checkpoint. ويقوم ال Alert Log File بتخزين معلومات حول ال Checkpoint كما يقوم بذلك مع ال Log Switch ولكن يجب وضع العامل "Parameter" التالي LOG_CHECKPONT_TO_ALERT في حالة True عكس الحالة الإفتراضية "Default" وهي False.

يمكن ان تحدث عملية ال Checkpoint عند:

- **حدوث عملية Log Switch.**
- **عند إغلاق ال Database بأي حالة عدا ال ABORT.**
- **عندما يحدد العامل "Parameter" FAST_START_MTTR_TARGET الذي يحدد كمية البيانات المتغيرة "Dirty Buffers" التي يستطيع ال DBWn كتابتها.**
- **عندما طلب ال DBA يمكن القيام بها بكتابة جملة SQL كما في المثال 4.1.**
- **عندما تكتب جملة ال SQL التالية: ALTER TABLESPACE OFFLINE NORMAL.**
- **عندما تكتب جملة ال SQL التالية: ALTER TABLESPACE READ ONLY.**
- **عندما تكتب جملة ال SQL التالية: ALTER TABLESPACE BEGIN BACKUP.**

مثال تطبيقي 4.1:

للقيام بعملية ال Checkpoint بواسطة ال SQL يجب كتابة التالي:

```
ALTER SYSTEM CHECKPOINT;
```

اذا تم تحديد العامل "Parameter"

```
FAST_START_MTTR_TARGET = 400
```

يدل على عملية ال Recovery يجب أن لا يزيد وقتها عن 400 ثانية حيث أن من ضمن عملية ال Recovery عملية ال Checkpoint وعملية ال Log Switch.

يمكن أن يتم طلب عملية ال Log Switch ايضاً بواسطة ال SQL عبر التالي:

```
ALTER SYSTEM SWITCH LOGFILE;
```

MAINTAINING REDO LOG FILES

سوف نتطرق الى عدد من المهام التي يمكن اجرائها لل Redo Log Files لحمايتها أو لتعديلها.

:MULTIPLEXING

لحماية ال Database يجب القيام بعملية ال Multiplex على ال Redo Log Files ، بحيث يقوم ال LGWR بكتابة ذات البيانات على النسخ المتشابه في نفس الوقت. كل النسخ المتشابه تكون ضمن مجموعة "Group" و يطلق على النسخ المتشابه اسم Members. يفضل فصل النسخ المتشابه عن بعض وابقائها في موقع مختلف أو على قرص صلب مختلف (أو أي وسيلة لحفظ البيانات) كما في الرسم 4.1 يظهرأن الأعضاء (Member1 , Member2) في ذات المجموعة على قرصين مختلفين (Disk1, Disk2).
في حال عدم توفر أي عضو "Member" من أي مجموعة لنقل البيانات إليه يتم اغلاق ال Database وتبدأ عملية Recovery لل Instance.
يمكن تكوين أكثر من مجموعة من مجموعات ال Redo Log Files عند تكوين ال Database أو يمكن اضافة مجموعة أخرى لاحقاً.

نقاط مهمة في عملية ال Multiplex:

- **ينصح بتفريق الأعضاء ضمن المجموعة الواحدة الى مواقع أو اقراص صلبة مختلفة، لكي لا يحدث أغلاق لل Database في حال تعطل القرص الصلب الذي يحوي الملفات.**
- **ينصح بتفريق ال Online Redo Log Files عن ال Archived Log Files الى أقراص مختلفة أو مواقع مختلفة لكي لا يحدث تداخل بين عمل ال LGWR و ال ARCn.**
- **ينصح بتفريق ملفات ال Redo Log و ملفات ال Data Files الى مواقع أو اقراص مختلفة، لكي لا يحدث تداخل بين ال DBWn وال LGWR.**

:SIZING ONLINE REDO LOG FILES

أصغر حجم يمكن ان يكون علية ال Redo Log File هو 50 KB ، أما اكبر حجم هو أكبر حجم يسمح به نظام التشغيل (مثل ويندوز أو UNIX).
يمكن لل DBA تحديد حجم ال Redo Log Files بالأخذ بالمعطيات التالية:

- **الحجم المتوفر على القرص الصلب أو أي Storage Device.**
- **عدد مرات ال Log Switch و ال Checkpoint التي يمكن أن تحدث.**
- **الحجم المتوقع لل Redo Entries والذي يعتمد على العمليات التي تجري على ال Database والتي تؤدي الى تغيير البيانات.**

ADDING ONLINE GROUP:

في حال وجد ال DBA أن ال Database بحاجة الى مجموعة جديدة ، يمكن تكوينها كما في المثال 4.2:

مثال تطبيقي 4.2:

```
ALTER DATABASE ADD LOGFILE GROUP 4
('...\oradata\db01\log11.log' , '...\oradata\db02\log12.log') SIZE 5M;
```

ADDING ONLINE MEMBER:

يمكن اضافة أعضاء جدد للمجموعات الموجودة باستخدام القاعدة التالية:

```
ALTER DATABASE [database's name] ADD LOGFILE MEMBER
['filename' [REUSE] , 'filename' [REUSE], ....]
TO [ GROUP number | ('filename', 'filename', ...);
```

في حالة وجود اسم العضو على الجهاز وبنفس الحجم (أي الملف موجود مسبقاً) يجب استخدام كلمة REUSE اما بالنسبة الى الجملة الأخيرة فلديك خياران إما أن تذكر رقم المجموعة أو اسماء الأعضاء الحاليين في المجموعة التي تريد اضافة عضو جديد لها.

ملاحظة: يمكن اضافة عضو جديد باستخدام ال Storage Manager في ال Console.

مثال تطبيقي 4.3:

```
ALTER DATABASE ADD LOGFILE MEMBER
'...\oradata\db04\log41.log' TO GROUP 1,
'...\oradata\db04\log42.log' TO GROUP 2,
'...\oradata\db04\log43.log' TO GROUP 3,
'...\oradata\db04\log44.log' TO GROUP 4;
```

أو بالطريقة الثانية:

```
ALTER DATABASE ADD LOGFILE MEMBER
'...\oradata\db04\log14.log'
TO ('...\oradata\db01\log11.log' , '...\oradata\db02\log12.log' ,
'...\oradata\db03\log13.log');
```

أو عبر ال Console باتباع الخطوات التالية:

- ادخل الى ال Console عبر Standalone.
- اضغط على اسم ال Database لكي تظهر نافذة تطلب من اسم المستخدم و كلمة السر.
- أدخل اسم المستخدم و كلمة السر مع التاكد من اختيار SYSDBA عوضاً عن Normal.
- اضغط على ال Storage Manager.
- اضغط على Redo Log Groups.
- اكتب اسم العضو الجديد ثم اضغط Apply.

:DROPPING ONLINE LOG GROUP

اذا أردت زيادة حجم أحدى مجموعات ال Redo Log يمكن ازالة المجموعة وتكوين مجموعة جديدة بحجم أكبر، حيث تستطيع ازالة مجموعة من مجموعات ال Redo Log باتباع القاعدة التالية:

```
ALTER DATABASE [database' name]
DROP LOGFILE [GROUP number | ('filename',....);
```

يجب أن لا تكون المجموعة المزالة هي المجموعة الحالية "Current" أو ان تكون في وضعية Active بل يجب أن تكون في وضعية Inactive. لا يسمح الأوراكل بحذف المجموعة اذا كان ذلك سوف يؤدي الى بقاء مجموعة واحدة فقط ، اذ أن الأوراكل يحتاج على الأقل الى مجموعتين من مجموعات ال Redo Log Files. باتباع القاعدة السابقة يمكن حذف المجموعة من تعريف الأوراكل فقط ، اما ملفات ال Redo Log الحقيقية الموجودة في الموقع المحدد (كما في المثال 4.3: ملف log14.log) لا تمسح ويجب حذفها بطريقة يدوية (كما في أوامر نظام التشغيل، في الويندوز استخدم أمر Delete).

مثال تطبيقي 4.4:

```
ALTER DATABASE DROP LOGFILE GROUP 4;
```

أو بذكر أسم أحد أعضاء المجموعة كما في المثال 4.3.

يمكن أيضا تحويل حالة ال Redo Log File الى حالة ال Inactive بواسطة الجملة التالية:

```
ALTER SYSTEM SWITCH LOGFILE;
```

سوف نتطرق الى الحلات التي يمكن أن تكون عليها المجموعة أو العضو بعد قليل.

:DROPPING ONLINE LOG MEMBER

كما يمكن حذف مجموعة يمكن حذف عضو (أو أكثر) من المجموعة ولكن يجب أن يبقى على الأقل عضو فعال "Valid" في المجموعة وإلا فإن أوراكل لن يسمح بعملية الحذف. كما في حذف المجموعة يجب أن لا يكون العضو هو العضو الحالي "Current" و الملفات الفيزيائية لا تحذف من مكان تواجدها. يوجد حالة أخرى لا يسمح الأوراكل بحذف العضو، ذلك عندما يكون ال Database في حالة ال Archivelog ولم يتم عمل Archive للمجوعة التي ينتمى لها العضو.

يمكن حذف عضو باتباع القاعدة التالية:

```
ALTER DATABASE [database's name]
DROP LOG MEMBER 'filename','filename',...
```

مثال تطبيقي 4.5:

ALTER DATABASE DROP LOG MEMBER '...\oradata\db04\log4.log';

أو عبر ال Console باتباع الخطوات التالية:

- ادخل الى ال Console عبر Standalone.
- اضغط على اسم ال Database لكي تظهر نافذة تطلب من اسم المستخدم و كلمة السر.
- أدخل اسم المستخدم و كلمة السر مع التاكد من اختيار SYSDBA عوضاً عن Normal.
- اضغط على ال Storage Manager.
- اضغط على Redo Log Groups ، ثم اختر العضو المراد حذفه واضعط على رسم سلة المهملات.

RENAMING LOG MEMBER:

اذا كنت تريد نقل عضو "Member" من موقع الى أخر أو تريد تغيير اسم العضو يجب اتباع التالي:

- اغلق ال Database.
- انسخ الملف الفيزيائي (الخاص بالعضو) الى الموقع الجديد (لنقل الملف) ، أوقم بتغيير اسم الملف (لتغيير اسم الملف).
- قم بتشغيل ال Database في حالة ال MOUNT.
- اكتب الجملة التالية

ALTER DATABASE RENAME FILE 'old Filename' TO 'newFilename';

- قم بتشغيل ال Database وقم بعملية Backup لل Control File على اعتبار أن محتوياته تغيرت.

CLEARING ONLINE REDO LOG FILES:

في حال حدوث فساد للبيانات الموجودة داخل ملفات ال Redo Log يمكن تنظيف الملفات باستخدام القاعدة التالية:

ALTER DATABASE [database's name]
CLEAR [UNARCHIVED] LOGFILE GROUP number | 'filename' ,.....;

يمكن العدول عن عملية حذف ثم اضافة ملف Redo Log بالقيام بتنظيف محتويات الملف لما توفره العملية من تسهيلات، اذ يمكن تنظيف مجموعة من مجموعات ال Redo Log حتى ولو وجد مجموعتيين فقط وبداخل كل مجموعة عضو واحد فقط ، ويمكن تنظيف المجموعة اذا كانت ال Database في حالة ال Archivelog ولم يحدث عملية Archive للمجموعة ولكن يجب استخدام كلمة UNARCHIVED في القاعدة.

مثال تطبيقي 4.6:

لتنظيف مجموعة من مجموعات ال Redo Log اكتب التالي:

```
ALTER DATABASE CLEAR LOGFILE GROUP 3;
```

أو بذكر اسماء أعضاء المجموعة عوضاً عن المجموعة ورقمها كما جاء سابقاً.

USING OMF:

لتسهيل عملية ادارة ال Redo Log Files ينصح باستخدام ال OMF التي جاء شرحها سابقاً. بواسطة ال OMF تكوين عملية تكوين مجموعة جديدة أو حذف مجموعة أسهل و يقوم الأوراكل بحذف الملفات الفيزيائية الموجودة في موقع التخزين بشكل أتوتوماتيكي عند حذف أي مجموعة.
لاستخدام طريقة ال OMF ، يجب تحديد العامل DB_CREATE_ONLINE_LOG_DEST_n في ال Initialization Parameter File وللقيام بعملية ال Multiplex على ال Redo Log Files يجب استبدال ال n في العامل "Parameter " برقم من 1 الى 5.

مثال تطبيقي 4.7:

لحذف مجموعة تعتمد على نظام ال OMF يجب كتابة:

```
ALTER DATABASE DROP LOGFILE GROUP 3;
```

أما لإضافة مجموعة يجب كتابة:

```
ALTER DATABASE ADD LOGFILE;
```

حيث يقوم الأوراكل بتحديد رقم المجموعة واختيار اسماء الأعضاء أوتوماتيكياً.

للقيام بعملية ال Multiplex يجب تحديد التالي في ال Initialization Parameter File:

```
DB_CREATE_ONLINE_LOG_DEST_1 = '...\oradata\db01'

DB_CREATE_ONLINE_LOG_DEST_2 = '...\oradata\db02'

DB_CREATE_ONLINE_LOG_DEST_3 = '...\oradata\db03'
```

:QUERYING LOG FILE INFORMATION

يمكن الحصول على معلومات حول المجموعات و الأعضاء من خلال:

- **V$LOG**
- **V$LOGFILE**

:V$LOG

تحتوى على بيانات حول المجموعات وأحجامها والحالة التي هي فيها، يوجد 7 حالات يمكن أن تتواجد فيها المجموعة وهم:

- **UNUSED: تدل على أن المجموعة لم تستخدم على الأطلاق في تخزين البيانات ، غالباً ما يكون هذا النوع للمجموعات المضافة حديثاً.**
- **CURRENT: تدل على المجموعة الحالية التي يستخدمها ال LGWR في نقل بيانات ال Redo Log Buffer.**
- **ACTIVE: تدل على أن المجموعة جاهزة لعملية ال Recovery.**
- **CLEARING: تدل أن المجموعة في حالة تنظيف لبياناتها بعدما طلب ال DBA ذلك بواسطة جملة ال SQL التي تطرقنا لها قبل قليل وبعد ان تنظف المجموعة تتحول الحالة الى UNUSED.**
- **CLEARING_CURRENT: تدل أن المجموعة تم تنظيفها بعد طلب جملة ال SQL.**
- **INACTIVE: تدل على أن ال Database ليس بحاجة الى المجموعة ولا تلزم في عملية ال Recovery ، وهذا هو النوع الذي يمكن حذف.**

:V$LOGFILE

تحتوى على بيانات حول أعضاء المجموعات. يوجد 4 حالات يمكن أن يتواجد فيها العضو وهم:

- **INVALID: تدل على أن ملف العضو لا يعمل.**
- **STALE: تدل على أن بيانات العضو غير مكتملة.**
- **DELETED: تدل على أن العضو لا يستخدم.**
- **فارغ "Blank": تدل على أن العضو يتم استخدامه.**

مثال تطبيقي 4.8:

```
SELECT GROUP# , SEQUENCE#, BYTES, MEMBERS, STATUS
FROM V$LOG;

SELECT GROUP#, STATUS, TYPE, MEMBER FROM V$LOGFILE;
```

ARCHIVED LOG FILES

من أهم الأمور التي يجب تحديدها من قبل ال DBA هو قرار وضع ال Database في وضعية ال Archivelog أو Noarchivelog. توجد ال Database في الحالة الأفتراضية "Default" في وضعية ال Noarchivelog ولكن وضعية ال Archivelog مفيدة جداً حيث أن من فوائدها تسهيل عملية ال Backup وتضمن استعادة "Recovery" البيانات التي حدث لها Commit. يمكن أن تتم عملية ال Archive لملفات ال Redo Log بطريقتيين هما: يدوية أو أوتوماتيكية والذي يتحكم بها عامل من عوامل ال Intialazation Parameter File هو LOG_ARCHIVE_START.
عندما تكون قيمة ال LOG_ARCHIVE_START تساوي True فإن عملية ال Archiving تحدث بشكل أتوتوماتيكي للمجموعة الممتلئة بواسطة ال ARCn بعد حدوث عملية Log Switch.
أما عندما تكون قيمة ال ARCHIVE_START_LOG تساوي False ، فأن عملية ال Archiving يجب أن تحدث بطريقة يدوية عبر استخدام جمل ال SQL.

ملاحظات:

- يتم تخزين بيانات في ال Control File اذا تمت عملية ال Archiving بنجاح.
- يمكن القيام بعملية Multiplex لملفات ال Archived Redo Log.
- لا يمكن لل Archived Redo Log أن تتبع نظام ال OMF.

تذكر: لا يمكن استخدام ال Redo Log File حتى تتم عملية ال Checkpoint ويتم عملية Archiving للملف (هذه الحالة تحدث عندما يكون الوضع Archivelog ، إما عند حالة Noarchivelog فقط عملية ال Checkpoint يجلب أن تحدث).

:QUERYING ARCHIVED REDO LOG

يمكن الحصول على بيانات حول عملية ال Archiving من V$INSTANCE أو من ال Console.

مثال تطبيقي 4.9:

```
SELECT ARCHIVER FROM V$INSTANCE;
```

أو عبر ال Console باتباع الخطوات التالية:

- ادخل الى ال Console عبر Standalone.
- اضغط على اسم ال Database لكي تظهر نافذة تطلب من اسم المستخدم و كلمة السر.
- أدخل اسم المستخدم و كلمة السر مع التاكد من اختيار SYSDBA عوضاً عن Normal.
- اضغط على ال Instance Manager ، ثم اضغط على Configuration.
- اضغط على كلمة ال Recovery.

الفصل الخامس

ملفات الداتا والتيبل سبيس

DATA FILES & TABLESPACES

TABLESPACE

كما عرفنا سابقاً أن ال Database تنقسم الى Logical و Physical بحيث تكون ال Tablespace هي أحدى مكونات ال Logical بالإضافة الى Segment و Extents وData Blocks. لا يمكن لل Tablespace العمل لأكثر من Database واحدة ولكن بالمقابل يمكن لل Tablespace أن تحتوي على أكثر من Data File.
يوجد نوعان من أنواع ال Tablespace هما:

- **SYSTEM TABLESPACE:** هي ال Tablespace التي تتكون مباشرة مع تكوين ال Database لأنها ضرورية لعمل ال Database ولا يمكن لل Database العمل بدونها. تحتوي على بيانات ال Data Dictionary وبرامج ال PL/SQL مثل Stored Units. تحتوي ايضاً على ال SYSTEM UNDO SEGMENT ويمكن أن تحتوي على البيانات "Data" ولكن لا يفضل جمع البيانات في ال System Tablespace بل في النوع الثاني من ال Tablespace.

- **NON-SYSTEM TABLESPACE:** لإضافة مرونة على التحكم بال Database ، ينصح باضافة Tablespaces الى ال Database لتخزين البيانات على مختلف أنواعها. من فوائد استعمالها أنها تقوم بفصل Undo Data ، عن Temporary Data ، عن Application Data أي أنها تفصل البيانات على حسب أختلافها ، ومن فوائد استعمالها ايضاً أن ال DBA يستطيع التحكم بحجم المساحة الممنوحة لكل مستخدم في ال Database.

CREATING TABLESPACES:

يمكن تكوين Tablespace باستخدام القاعدة التالية:

```
CREATE TABLESPACE tablespace's name
[DATAFILE C1]
[MINIMUM EXTENT number [K | M]]
[BLOCKSIZE number [K] ]
[LOGGING | NOLOGGING]
[DEFAULT C2]
[ONLINE | OFFLINE]
[PERMANENT | TEMPORARY]
```

ملاحظة: الكلمات التي تحتها خط في القاعدة هي ال Default.

أما ال الرموز ال C1، C2، فترمز الى جزء من القاعدة تم تخفيفه الى رموز لسهولة قراءة القاعدة:

C1= 'filename' [SIZE number [K | M] [REUSE] | REUSE] [AUTOEXTEND …]

C2= STORAGE (INITIAL number K|M NEXT number K|M MINEXTENTS number PCTINCREASE number MAXEXTENTS number)

سوف يأتي شرح بالتفصيل لعوامل ال C2 المبينة (Initial, Next, …) و جملة ال AUTOEXTEND.

أما أوامر القاعدة فهي :

- **[MINIMUM EXTENT]: تحدد حجم جميع ال Extents الذين ينتمون الى ال Tablespace بحيث يكون الحجم مضاعفات الرقم الموجود بجانب الجملة، يعني اذا كان الرقم هو 4 فيكون أحجام ال Extents مضاعفات هذا الرقم (4, 8, 16,...).**
- **[BLOCKSIZE]: لتحديد حجم ال Block Size تعرفنا الى العامل DB_BLOCK_SIZE والذي يحدد الحجم المسمى بال Standard ولكن يسمح الأوراكل بوجود أربعة أحجام أخرى لل Block Size تسمى Non-Standard في حالة أن حجم ال Standard لا يناسب ال Tablespace وعندئذ يمكن تحديد ال Block Size بواسطة هذه الجملة ويكون حجم ال Non-Standard بين 2KB الى 32KB.**
- **[LOGGING]: تحدد أن جميع عناصر ال Tablespace من Tables و Indexes وغيرها يجب أن يتم كتابة المتغيرات التي تطرأ عليها الى ال Redo Log Files.**
- **[NOLOGGING]: تحدد أن جميع عناصر ال Tablespace من Tables و Indexes وغيرها يجب أن لا يتم كتابة المتغيرات التي تطرأ عليها الى ال Redo Log Files.**
- **[DEFAULT C2]: تحدد عوامل خاصة تسمى عوامل التخزين "Storage Parameters" التي سوف نتطرق لها لاحقاً.**
- **[OFFLINE]: يوجد عدد من الحالات التي يمكن أن تكون عليها ال Tablespace بعد تكوينها والحالة تحدد أنها غير جاهزة للاستخدام (ضد Online).**
- **[PERMANENT]: تحدد أن نوعية ال Tablespace من النوع الذي يحمل بيانات دائمة.**
- **[TEMPORARY]: تحدد أن نوعية ال Tablespace من النوع الذي يحمل بيانات مؤقته.**

مثال تطبيقي 5.1:

لتكوين ال Tablespace باستخدام جملة ال SQL يمكن كتابة:

```
CREATE TABLESPACE user_data
DATAFILE '...\oradata\db01\userdata01.dbf' SIZE 50M
AUTOEXTEND ON NEXT 5M MAXSIZE 100M
LOGGING
OFFLINE
TEMPORARY
DEFAULT STORAGE (INITIAL 1M  NEXT 1M  MINEXTENTS 2
PCTINCREASE 0 MAXEXTENTS
999)
;
```

أو عبر ال Console باتباع الخطوات التالية:

- **ادخل الى ال Console عبر Standalone.**
- **اضغط على اسم ال Database لكي تظهر نافذة تطلب من اسم المستخدم و كلمة السر.**
- **أدخل اسم المستخدم و كلمة السر مع التاكد من اختيار SYSDBA عوضاً عن Normal.**
- **اضغط على ال Storage Manager.**
- **اضغط على Tablespace ثم بواسطة الضغط بالزر اليمين للفأرة على ال Tablespace تظهر أوامر ، اختر Create.**
- **تظهر نافذة جديدة ، ادخل البيانات المطلوبة مثل اسم ال Tablespace وال Data Files التابعين لل Tablespace وباقي البيانات ، ثم اضغط CREATE.**

MANAGING TABLESPACES SPACE

عندما يتم تخصيص مساحة لأي عنصر داخل ال Tablespace يكون لكل عنصر Segment تحوي على مجموعة من ال Extents، ويمكن ادارة مساحة ال Extent بطريقتين هما:
Locally Managed Tablespace وDictionary Managed Tablespace

ملاحظة: لا يمكن تغيير طريقة إدارة ال Extents بعد تحديدها.

:LOCALLY MANAGED TABLESPACE

تتبع ال Tablespace هذه الطريقة عندما يتم تخزين معلومات ال Extents في ال Data Files التابعة لل Tablespace. يتم تخزين البيانات على شكل Bitmap (خريطة بايتات) والتي تدل على ال Free Blocks و ال Used Blocks في ال Extents، وعندا يحدث أي تعديل على ال Blocks يقوم الأوراكل بتجديد بيانات الBitmap لتوافق التغيرات التي طرأت.

من فوائد استخدام طريقة ال Locally Managed:

- تخفف الضغط على ال Data Dictionary.
- حجم ال Extent يمكن أن يقدر بشكل أوتوماتيكي و جميع ال Extents تكون بنفس الحجم.
- عدم الحاجة الى جمع المساحات الخالية الصغيرة الضائعة التي تنشئ بين البيانات نتيجة للتغيرات التي تحدث وتعيدها الى المساحة الخالية الرئيسية وتسمى هذه العملية Coalescing Space (بمعنى أخر أنها تمنع حدوث عمليات عديدة تسبب ضغط على ال Database).

لتكوين Tablespaces تتبع هذه الطريقة ، يجب اضافة الجملة التالية الى قاعدة :CREATE TABLESPACE

```
EXTENT MANAGEMENT LOCAL
[AUTOALLOCATE | UNIFORM [SIZE number [K|M] ] ]
```

نقاط مهمة:

1- لا يمكن استخدام جملة ال [DEFAULT C2]

2- لا يمكن استخدام كلمة ال TEMPORARY.

3- لا يمكن استخدام جملة ال MINIMUM EXTENT.

أما أوامر الجملة فهي :

- AUTOALLOCATE: يقوم الأوراكل بتحديد حجم ال Extents ولا يستطيع الDBA فعل ذلك.
- UNIFORM: يتم تحديد حجم ال Extents من قبل ال DBA.

مثال تطبيقي 5.2:

يمكن تكوين Tablespace تتبع طريقة ال Locally Managed بكتابة التالي:

```
CREATE TABLESPACE app_data
DATAFILE '...\oradata\db01\appdata01.dbf' SIZE 250M
EXTENT MANAGEMENT LOCAL UNIFORM SIZE 256K;
```

أو اذا تم كتابة التالي فقط ، فتعتبر Locally Managed ويؤخذ ال Default وهو Autoallocate:

```
CREATE TABLESPACE app_data
DATAFILE '...\oradata\db01\appdata01.dbf' SIZE 250M;
```

DICTIONARY MANAGED TABLESPACE:

تتبع ال Tablespace هذه الطريقة عندما يتم تخزين معلومات ال Extents في ال Data Dictionary. من فوائد استخدام طريقة ال Dictionary Managed انها توفر مرونة في تحديد حجم كل Segment على حده باستخدام جملة ال [DEFAULT C2]. ولكن في المقابل تلزم الحاجة الى جمع المساحات الخالية الصغيرة الضائعة التي تنشئ بين البيانات نتيجة للتغيرات التي تحدث و تعيدها الى المساحة الخالية الرئيسية ، أي حدوث عمليات قد تسبب مزيد من الضغط على ال Database.

لتكوين Tablespaces تتبع هذه الطريقة ، يجب اضافة الجملة التالية الى قاعدة :CREATE TABLESPACE

EXTENT MANAGEMENT DICTIONARY [DEFAULT C2]

مثال تطبيقي 5.3:

```
CREATE TABLESPACE app_data
DATAFILE '...\oradata\db01\appdata01.dbf' SIZE 250M
EXTENT MANAGEMENT DICTIONARY
DEFAULT STORAGE (INITIAL 1M NEXT 1M] ;
```

ملاحظة: يمكن تغيير قيم العوامل ضمن الجملة [DEFAULT C2] أو جملة ال MINIMUM EXTENT باستخدام الأمر ALTER TABLESPACE ، أو عبر ال Console.

مثال تطبيقي 5.4:

لتغيير جملة ال Minimum Extent:

```
ALTER TABLESPACE app_data MINIMUM EXTENT 2M;
```

لتغيير جملة ال ...Default Storage:

```
ALTER TABLESPACE user_data
DEFAULT STORAGE (
INITIAL 2M
NEXT 2M
MINEXTENTS 1500);
```

أو عبر ال Console باتباع الخطوات التالية:

- ادخل الى ال Console عبر Standalone.
- اضغط على اسم ال Database لكي تظهر نافذة تطلب من اسم المستخدم و كلمة السر.
- أدخل اسم المستخدم و كلمة السر مع التاكد من اختيار SYSDBA عوضاً عن Normal.
- اضغط على ال Storage Manager.
- اضغط على Tablespace ثم الضغط بالزر اليمين للفأرة على اسم ال Tablespace المراد تغير خواصها لتظهر قائمة أوامر، أختر من الأوامر View\Edit Details.
- تظهر نافذة جديدة ، اضغط على كلمة Storage ثم أدخل التغيرات المطلوبة، ثم اضغط Apply.

تذكر: أنه لا يمكن تغيير خواص ال Storage لل Tablespace التي تتبع طريقة ال Locally Managed.

NON-SYSTEM TABLESPACES

يوجد أكثر من Non-System Tablespaces ولكل منها وظائف مختلفة في ال Database ، منها Permanent Tablespace و Undo Tablespace و Temporary Tablespace.
أما Permanent Tablespace فوظائفها تخزين البيانات "Data" لفصل بيانات ال Database عن بيانات ال Data Dictionary و لسهولة التحكم في البيانات بحيث اذا تم حذف ال Tablespace الخاص بها لا يؤثر ذلك على عمل ال Data Dictionary و لتخفيف الضغط على ال System Tablespace وال Database بشكل عام.

:UNDO TABLESPACE

تستخدم لحفظ ال Undo Segment ولا يمكن أن تحتوي على غيرها. يتم إدارة ال Extents بطريقة ال Locally Managed و يمكن تكوينها باتباع القاعدة التالية (أو تكوينها عند تكوين ال Database باستخدام CREATE DATABASE):

```
CREATE UNDO TABLESPACE tablespace_name
[DATAFILE C1]
EXTENT MANAGEMENT LOCAL
[AUTOALLOCATE | UNIFORM [SIZE number [K|M] ] ]
```

ملاحظة: تستخدم ال Undo Tablespace في ال Automatic Undo Management التي سوف نتطرق لها في فصل أخر مع ال Undo Segment.

مثال تطبيقي 5.5:

```
CREATE UNDO TABLESPACE undo01
DATAFILE '...\oradata\db01\undo101.dbf' SIZE 50M;
```

أو عبر ال Console باتباع الخطوات التالية:

- **ادخل الى ال Console عبر Standalone.**
- **اضغط على اسم ال Database لكي تظهر نافذة تطلب من اسم المستخدم و كلمة السر.**
- **أدخل اسم المستخدم وكلمة السر مع التاكد من اختيار SYSDBA عوضاً عن Normal.**
- **اضغط على ال Storage Manager.**
- **اضغط على Tablespace ثم الضغط بالزر اليمين للفأرة على اسم ال Tablespace المراد تغير خواصها لتظهر قائمة أوامر، أختر من الأوامر Create.**
- **تظهر نافذة جديدة ، أختر Undo عوضاً عن Permanent ثم ادخل البيانات المطلوبة مثل اسم ال Tablespace ثم أضغط Create.**

تذكر: بيانات ال Undo Segment تخزن في ال Control File و ال Alert Log File و ال Initialization Parameter File.

:TEMPORARY TABLESPACE

توفر المساحة اللازمة لعمليات ال Sort المختلفة الناتجة عن أوامرعدة مثل **Order By** أو**Group By**. بمعنى أخر أنها تخزن البيانات المؤقتة "Temporary Data"، وتحوي ما يعرف ب **Sort Segment** والتي تتكون في ال Tablespace عند أول عملية Sort تطلب من ال Instance. يمكن إدارة ال Extents بالطريقتين و لكن ينصح باستخدام Locally Managed.

ملاحظة: تستطيع ال Sort Segment التوسع في المساحة بحجز المزيد من ال Extents لكي يتم تغطية المساحة اللازمة لعمليات ال Sort المختلفة.

يمكن تكوين ال Temporary Tablespace باستخدام جملة **CREATE TABLESPACE** مع إضافة كلمة **TEMPORARY** في الجملة و لكن يفضل استخدام القاعدة التالية لتكوين Locally Managed:

```
CREATE TEMPORARY TABLESPACE temp's name
TEMPFILE 'filename'
EXTENT MANAGEMENT LOCAL
[AUTOALLOCATE | UNIFORM [SIZE number [K|M] ] ]
```

مثال تطبيقي 5.6:

```
CREATE TEMPORARY TABLESPACE temp01
TEMPFILE '...\oradata\db05\temp01.dbf ' SIZE 200M
EXTENT MANAGEMENT LOCAL UNIFORM SIZE 10M;
```


أو عبر ال Console باتباع الخطوات التالية:

- ادخل الى ال Console عبر Standalone.
- اضغط على اسم ال Database لكي تظهر نافذة تطلب من اسم المستخدم و كلمة السر.
- أدخل اسم المستخدم وكلمة السر مع التاكد من اختيار SYSDBA عوضاً عن Normal.
- اضغط على ال Storage Manager.
- اضغط على Tablespace ثم بواسطة الضغط بالزر اليمين للفأرة على ال Tablespace تظهر أوامر ، اختر Create.
- تظهر نافذة جديدة ، أختر الخيار Temporary عوضاً عن Permanent ثم ادخل البيانات المطلوبة مثل اسم ال Tablespace و أدخل البيانات التابعة لقسم ال Storage مثل ال Size.
- ثم أضغط Create.

يعتبر TEMPFILE (الذي استخدم في القاعدة) متطابقاً لل DATAFILE عدا في الاختلافات التالية:

- لا يمكن وضعه في حالة القراءة فقط Read-Only.
- لا يمكن تغيير اسم الملف.
- لا يمكن أن يتم له عملية Recovery.
- لا يمكن تكوينه باستخدام ال **ALTER DATABASE**.
- دائماً في وضعية ال **NOLOGGING**.
- لا يمكن استخدامها مع جملة ال **CREATE CONTROLFILE**.

:Default Temporary Tablespaces

باستخدام قاعدة **CREATE DATABASE** يمكن تكوين **Default Temporary Tablespace** والتي تحمي ال **Database** من استخدام ال **System Tablespace** عوضاً عن ال **Temporary** في تنفيذ عمليات ال **Sorts** والتي قد تؤدي الى مشاكل عدة في ال **System Tablespace** وازدياد نسبة ال **Fragmentation** داخل ال **System Tablespace**.
في حال لم يتم تكوين ال **Temporary Tablespace** مع تكوين ال **Database** وتم تكوينها بعد ذلك ، يمكن تحديد أنها ال **Default Temporary** بواسطة أمر **ALTER DATABASE**، وجميع المستخدمين الذين كانوا يستخدمون ال **System Tablespace** لعمليات ال **Sorts** يتم تحويلهم بشكل أوتوماتيكي الى ال **Default Temporary Tablespace**.

ملاحظة: اذا اراد ال **DBA** تغيير ال **Default Temporary Tablespace** الى **Temporary Tablespace** أخرى لتكون هي ال **Default** ، يمكن استخدام **ALTER DATABASE**.

ملاحظة2: اذا تم تكوين ال **Default Temporary Tablespace** بواسطة **CREATE DATABASE** فإنها تتبع طريقة ال **Locally Managed**.

قيود ال Default Temporary Tablespace:

- لا يمكن حذف ال **Default Temporary Tablespace** إلا بعد تحويلها الى **Temporary Tablespace** أخرى بواسطة ال **ALTER DATABASE**.
- لا يمكن تحويل ال **Default Temporary** الى **Permanent**.
- لا يمكن وضعها في حالة ال **Offline**.

مثال تطبيقي 5.7:

لتعريف ال **Temporary Tablespace** كنوعية ال **Default** أو لتغيير ال **Default Temporary** الى أخرى يجب كتابة التالي:

```
ALTER DATABASE DEFAULT TEMPORARY TABLESPACE temp02;
```

أو عبر ال **Console** باتباع الخطوات التالية:

- ادخل الى ال **Console** عبر **Standalone**.
- اضغط على اسم ال **Database** لكي تظهر نافذة تطلب من اسم المستخدم و كلمة السر.
- أدخل اسم المستخدم وكلمة السر مع التاكد من اختيار **SYSDBA** عوضاً عن **Normal**.
- اضغط على ال **Storage Manager**.
- اضغط على **Tablespace** ثم بواسطة الضغط بالزر اليمين للفأرة على ال **Tablespace** تظهر أوامر ، اختر **Create**.
- تظهر نافذة جديدة ، أختر الخيار **Temporary** عوضاً عن **Permanent** ثم ادخل البيانات المطلوبة مثل اسم ال **Tablespace** وأدخل البيانات التابعة لقسم ال **Storage** مثل ال **Size**.
- يجت تحديد الخانة **Set as Default Temporary Tablespace** ثم أضغط **Create**.

ALTERING A TABLESPACE

يمكن تعديل Tablespaces أو حذفهم بواسطة أمر ALTER TABLESPACE، بحيث يمكن تنفيذ بواسطة هذا الأمر: تغيير الحالة بين Offline و Online ، وضع ال Tablespace في وضعية القراءة فقط Read-Only ، حذف Tablespace ، تغيير حجم ال Tablespace، وغيرها من المهام.

:OFFLINE OR ONLINE

عندما تكون ال Tablespace في حالة Offline، لا يستطيع المستخدم استخراج البيانات المخزنة في ال Tablespace من ال Database. يمكن وضع ال Tablespace في حالة ال Offline لعدة اسباب منها أن ال DBA يرغب في اغلاق جزء من البيانات أو عمل عملية Backup أو Recovery على هذا الجزء دون إغلاق ال Database بشكل تام أو نقل أو تغيير اسم Data Files لل Tablespace دون إغلاق ال Database بشكل كامل.
عندما تتغير حالة ال Tablespace من Online الى Offline أو بالعكس، يتم تخزين معلومات حول العملية في ال Data Dictionary وفي ال Control File. يمكن لل Instance تحويل حالة ال Tablespace بين الحالتين عند حدوث أخطاء أو مشاكل مثل عدم المقدرة على نقل البيانات الى ال Tablespace.
لا يمكن وضع جميع ال Tablespaces في حالة ال Offline اذ يستثنى كل من ال System Tablespace و ال Default Temporary وال Tablespace التي تحوي Undo Segment في حالة Active.

يمكن التحويل بين الحالتين باستخدام القاعدة التالية:

ALTER TABLESPACE tablespace's name
ONLINE |
OFFLINE [NORMAL | TEMPORARY | IMMEDIATE| FOR RECOVER]

أوامر القاعدة:

- NORMAL: يتم كتابة جميع البيانات المتغيرة "Dirty Buffers" الخاصة بال Data Files في ذاكرة ال SGA الى ال Data Files ضمن ال Tablespace المراد إغلاقها.
- TEMPORARY: توفر عملية Checkpoint لجميع ال Online Data Files ضمن ال Tablespace المراد إغلاقها و لا توفر ذات العملية لل Offline Data Files.
- IMMEDIATE: لا تقوم بعملية Checkpoint على ال Data Files ضمن ال Tablespace المراد غلقها وبالتالي يجب القيام بعملية Recovery عند الرغبة بتحويلها الى Online.
- FOR RECOVER: لاستخدام ال Tablespace في عملية Recovery.

تذكر النقاط التالية:

- أن ال DBWn يعمل عندما يتم وضع ال Permanent Tablespace أو Temporary Tablespace في حالة ال Offline.

- أن وضع ال Tablespace في حالة ال Offline Normal يؤدي الى حدوث عملية ال Checkpoint.

- أنه يمكن وضع ال Tablespace في احدى الحالتين عند تكوينها بواسطة CREATE TABLESPACE.

مثال تطبيقي 5.8:

لوضع ال Tablespace في حالة ال Online يجب كتابة:

ALTER TABLESPACE user_data ONLINE;

لوضع ال Tablespace في حالة ال Offline يمكن كتابة:

ALTER TABLESPACE user_data OFFLINE;

ALTER TABLESPACE user_data OFFLINE IMMEDIATE;

ALTER TABLESPACE user_data OFFLINE TEMPORARY;

أو عبر ال Console باتباع الخطوات التالية:

- **ادخل الى ال Console عبر Standalone.**
- **اضغط على اسم ال Database لكي تظهر نافذة تطلب من اسم المستخدم و كلمة السر.**
- **أدخل اسم المستخدم وكلمة السر مع التاكد من اختيار SYSDBA عوضاً عن Normal.**
- **اضغط على ال Storage Manager.**
- **اضغط على ال + بجانب كلمة Tablespace ليظهر قائمة بأسماء ال Tablespaces.**
- **اضغط على اسم ال Tablespace المراد تحويل حالتها.**
- **يظهر على النافذة اليمنى بعض خصائص ال Tablespace، اختر Offline وحالة الإغلاق من المجموعة التي بجانبها ثم اضغط على Apply.**

:READ-ONLY TABLESPACE

يمكن وضع ال Tablespace في حالة القراءة فقط "Read-Only" اذا رغب ال DBA بأن تتم قراءة بيانات ال Data Files ضمن ال Tablespace دون تغيير البيانات. لكي يستطيع ال DBA وضع ال Tablespace في حالة ال Read-Only يجب أن يكون جميع ال Data Files التابعين لل Tablespace في وضعية Online وإلا فأن العملية لن تنجح ، والعكس صحيح في حال تحويل ال Tablespace من Read-Only الى الحالة العادية Read-Write يحب توفر جميع ال Data Files في وضعية Online. يمكن وضع ال Tablespace في حالة Read-Only باستخدام القاعدة التالية:

ALTER TABLESPACE tablespace's name READ [ONLY | WRITE]

يمكن حذف عناصر (Tables, Indexes) من ال Read-Only Tablespace اذ أن حذف عناصر لا تكون بيانات جديدة في ال Tablespace ولا تغيير البيانات الموجودة إنما تغيير بيانات ال Data Dictionary. وفي حال كانت تجرى على ال Tablespace عدد من المهام "Transaction" عند عملية التحويل ، يتم منع حدوث مهام جديدة ويتم انجاز المهام التي تجري وإما أن تسجل البيانات الناتجة عن المهام عند حدوث Commit أو تلغى عند حدوث Rollback.

ملاحظة: لا يمكن وضع ال System Tablespace في حالة ال Read-Only.

مثال تطبيقي 5.9:

لوضع ال Tablespace في حالة ال Read-Only يجب كتابة:

```
ALTER TABLESPACE user_data READ ONLY;
```

لوضع ال Tablespace في الحالة العادية:

```
ALTER TABLESPACE user_data READ WRITE;
```

أو عبر ال Console باتباع الخطوات التالية:

- ادخل الى ال Console عبر Standalone.
- اضغط على اسم ال Database لكي تظهر نافذة تطلب من اسم المستخدم و كلمة السر.
- أدخل اسم المستخدم وكلمة السر مع التاكد من اختيار SYSDBA عوضاً عن Normal.
- اضغط على ال Storage Manager.
- اضغط على ال + بجانب كلمة Tablespace ليظهر قائمة بأسماء ال Tablespaces.
- اضغط على اسم ال Tablespace المراد تحويل حالتها.
- يظهر على النافذة اليمنى بعض خصائص ال Tablespace، ضع علامة في المربع بجانب كلمة Read Only ، ثم اضغط على Apply.

تذكر: أن وضع ال Tablespace في حالة ال Read-Only يؤدي الى عمل ال DBWn و حدوث عملية ال Checkpoint.

DROPPING TABLESPACES:

يمكن حذف ال Tablespace من ال Database باستخدام القاعدة التالية:

```
DROP TABLESPACE tablespace's name
[INCLUDING CONTENTS [AND DATAFILES] [CASCADE CONSTRAINTS]]
```

أوامر القاعدة:

- **INCLUDING CONTENTS:** اذا كانت ال Tablespace خالية لا تحوي أي بيانات "Data" فلا يشترط كتابة هذه الجملة ، ولكن اذا كانت تحوي على بيانات "Data" فيجب كتابة هذه الجملة.
- **AND DATAFILES:** كما تعرف أن لكل Tablespace مجموعة من ال Data Files التابعة لها تكون الجزء الفيزيائي (الملفات الحقيقة) على موقع التخزين (مثل القرص الصلب). عندما لا تكتب هذه الجملة لا تحذف ملفات ال Data Files من موقع التخزين والعكس صحيح اذ أن كتابة هذه الجملة تؤدي الى حذف ملفات ال Data Files بشكل أوتوماتيكي.
- **CASCADE CONSTRAINTS:** يجب استخدام هذه الجملة في حالة وجود ما يعرف ب Referential Integrity (Foreign Key) من Table توجد خارج ال Tablespace المراد حذفها مع Primary Key أو Unique Key لTable موجودة داخل ال Tablespace. في حال وجود هذا الرابط ولم يتم استخدام هذه الجملة ، تفشل عملية الحذف.

نقاط مهمة:

1- يمكن حذف ال Tablespace التي توجد في وضعية Read-Only.

2- ينصح بتحويل ال Tablespace الى حالة Offline قبل حذفها للتأكد من عدم وجود مهام أو عمليات تجري على ال Tablespace ، اذ بتحويلها الى Offline يتم وقف العمليات الجديدة وإنهاء المهمات الحالية كما شرحنا قبل قليل.

3- بعد حذف ال Tablespace يتم تعديل بيانات ال Control File لتتوافق مع التغيير الذي حدث في ال Tablespace.

تذكر: أن ال Control File يحوي معلومات حول أسماء ال Tablespaces وأسماء و مواقع ال Data Files.

مثال تطبيقي 5.10

لحذف Tablespace يمكن كتابة التالي:

```
DROP TABLESPACE user_data
INCLUDING CONTENTS AND DATAFILES;

DROP TABLESPACE user_data
INCLUDING CONTENTS CASCADE CONSTRAINT;

DROP TABLESPACE user_data
INCLUDING CONTENTS AND DATAFILES CASCADE CONSTRAINT;
```


أو عبر ال Console باتباع الخطوات التالية:

- **ادخل الى ال Console عبر Standalone.**
- **اضغط على اسم ال Database لكي تظهر نافذة تطلب من اسم المستخدم و كلمة السر.**
- **أدخل اسم المستخدم وكلمة السر مع التاكد من اختيار SYSDBA عوضاً عن Normal.**
- **اضغط على ال Storage Manager.**
- **اضغط على ال + بجانب كلمة Tablespace ليظهر قائمة بأسماء ال Tablespaces.**
- **اضغط على اسم ال Tablespace المراد حذفها.**
- **استخدم الزر اليمين للفأرة اتظهر لك قائمة، اختر Remove.**
- **أختر Yes لتأكيد الحذف.**

:RESIZING A TABLESPACE

يمكن زيادة حجم ال Tablespace بزيادة حجم ال Data Files التابعة لها أو إضافة Data Files جديدة لل Tablespace. يمكن زيادة حجم ال Data Files بطريقة يدوية أو أوتوماتيكية.

:Automatic Extension

يمكن استخدام جملة ال AUTOEXTEND لزيادة حجم ال Data Files بطريقة أوتوماتيكية أو لوقف هذه العملية. يمكن استخدام جملة ال AUTOEXTEND بعد تكوين ال Database أو ال Tablespace باستخدام القاعدة التالية:

```
ALTER DATABASE [database's name]
 DATAFILE 'filename' [SIZE number] [K | M] [REUSE]
[AUTOEXTEND OFF|ON [NEXT number [K|M]]
[MAXSIZE UNLIMITED | number [K |M]]
```

أوامر القاعدة:

- **AUTOEXTEND OFF: توقف عملية زيادة حجم الملفات بشكل أوتوماتيكي.**
- **AUTOEXTEND ON: لتشغيل عملية زيادة حجم الملفات بشكل أوتوماتيكي عند أمتلاء الملفات.**
- **NEXT: حجم الزيادة عند أمتلاء ال Data Files.**
- **MAXSIZE: لتحديد الحجم الأقصى الذي يمكن أن يصل له حجم ال Data Files. يمكن أن يحدد الحد الأقصى أو ان يكون غير محدد باستخدام UNLIMITED.**

تذكر: تستخدم جملة ال AUTOEXTEND في قاعدة CREATE DATABASE و CREATE TABLESPACE.

مثال تطبيقي 5.11

لزيادة حجم ال Data Files ضمن ال Tablespace بشكل أوتوماتيكي:

```
ALTER DATABASE DATAFILE '...\oradata\db01\userdata01.dbf'
AUTOEXTEND ON NEXT 10M MAXSIZE 250M;
```

أو عبر ال Console باتباع الخطوات التالية:

- **ادخل الى ال Console عبر Standalone.**
- **اضغط على اسم ال Database لكي تظهر نافذة تطلب من اسم المستخدم و كلمة السر.**
- **أدخل اسم المستخدم وكلمة السر مع التاكد من اختيار SYSDBA عوضاً عن Normal.**
- **اضغط على ال Storage Manager.**
- **اضغط على ال + بجانب كلمة Data Files ليظهر قائمة بأسماء ال Data Files.**
- **اضغط على اسم ال Data File المراد تكبيرها.**
- **اضغط على كلمة ال Storage لتدخل الى القسم الأخر .**
- **ضع علامة في المربع ثم أكتب قيم كل من Increment (NEXT) ، واختر إما Unlimited او قيمة Value ثم اضغط على Apply.**

:Manual Extension

يمكن زيادة أو انقاص حجم ال Data Files باستخدام القاعدة التالية:

```
ALTER DATABASE [database's name] DATAFILE 'filename', ['filename2'] …
 RESIZE number [K|M]
```

سؤال: لنفرض أنه يوجد لدينا Data File حجمه 100 MB وهو يحوي على 70 MB بيانات ، واراد ال DBA تصغير حجمه الى 60 MB ، فهل تنجح العملية ؟ و كم يكون حجم ال Data File بعد التصغير؟

جواب: نعم تنجح عملية التصغير ولكن يكون حجم ال Data File هو حجم البيانات فيه وهو 70 MB.

ملاحظة: يمكن تكبير أو تصغير حجم أكثر من ملف في نفس الوقت كما في المثال 5.12

مثال تطبيقي 5.12

لزيادة حجم ال Data Files ضمن ال Tablespace بشكل يدوي:

```
ALTER DATABASE DATAFILE '…\oradata\db01\userdata01.dbf',
'….\oradata\db01\userdata02.dbf'
RESIZE 250M;
```

أو عبر ال Console باتباع الخطوات التالية:

- **ادخل الى ال Console عبر Standalone.**
- **اضغط على اسم ال Database لكي تظهر نافذة تطلب من اسم المستخدم و كلمة السر.**
- **أدخل اسم المستخدم وكلمة السر مع التاكد من اختيار SYSDBA عوضاً عن Normal.**
- **اضغط على ال Storage Manager.**
- **اضغط على ال + بجانب كلمة Data Files ليظهر قائمة بأسماء ال Data Files.**
- **اضغط على اسم ال Data File المراد تكبيرها أو تصغيرها.**
- **ادخل الحجم الجديد في خانة File Size ثم اضغط Apply.**

:Adding Data File

يمكن اضافة Data Files الى ال Tablespace لتكبير حجمها عبر القاعدة:

```
ALTER TABLESPACE tablespace's name ADD DATAFILE
'filename' [SIZE number [K|M]] [REUSE] [AUTOEXTEND …] , ….
```

ملاحظة: يمكن اضافة أكثر من ملف في نفس الوقت. استخدم فاصلة في أخر القاعدة وأعد كتابة السطر الثاني. كما في المثال 5.13.

مثال تطبيقي 5.13:

لاضافة Data Files الى ال Tablespace:

```
ALTER TABLESPACE user_data ADD DATAFILE
'…\oradata\db01\userdata01.dbf' SIZE 200M
AUTOEXTEND ON NEXT 10M MAXSIZE 250M,
'…\oradata\db01\userdata02.dbf' SIZE 200M
AUTOEXTEND ON NEXT 10M MAXSIZE 250M;
```

أو عبر ال Console باتباع الخطوات التالية:

- ادخل الى ال Console عبر Standalone.
- اضغط على اسم ال Database لكي تظهر نافذة تطلب من اسم المستخدم و كلمة السر.
- أدخل اسم المستخدم وكلمة السر مع التاكد من اختيار SYSDBA عوضاً عن Normal.
- اضغط على ال Storage Manager.
- اضغط على ال + بجانب كلمة Tablespace ليظهر قائمة بأسماء ال Tablespaces.
- اضغط على اسم ال Tablespace المراد اضافة Data File لها.
- استخدم الزر اليمين للفأرة اتظهر لك قائمة، Add Datafile.
- تظهر لك نافذة جديدة ، ادخل البيانات مثل اسم الملف ثم اضغط Create.

:MOVING DATAFILES

يمكن نقل ال Data Files من موقع الى أخر باستخدام إما ALTER TABLESPACE أو ALTER DATABASE.

باستخدام Alter Tablespace:

يمكن نقل ملفات ال Data Files باستخدام القاعدة التالية:

```
ALTER TABLESPACE tablespace's name
RENAME DATAFILE 'filename' , ['filename']
TO 'filename' , ['filename'];
```

طريقة التنفيذ:

- ضع ال Tablespace في حالة Offline.
- نسخ ملف ال Data File الى المواقع الجديدة (القيام بعملية النسخ كما في نظام التشغيل ، اي في الويندوز يستخدم الفأرة مع COPY و PASTE).
- تنفيذ القاعدة السابقة.
- تحويل ال Tablespace الى الحالة Online
- يمكن حذف ال Data Files الموجودة في الموقع القديم كما في نظام التشغيل.

شروط الاستخدام:

- يجب أن تكون ال Tablespace في حالة ال Offline.
- يجب تواجد ملفات ال Data Files المراد نقلها في الموقع المحدد في القاعدة.
- يجب أن تتطابق اسماء ال Data Files المنقولة مع اسماء ال Data Files المخزنة في ال Control File.
- لا يمكن نقل Data Files التابعيين ل System Tablespace.

مثال تطبيقي 5.14:

بعد تنفيذ النقطتين الأولى و الثانية من طريقة التنفيذ:

```
ALTER TABLESPACE app_data RENAME DATAFILE
'...\oradata\db01\app01.dbf' , '...\oradata\db01\app02.dbf'
TO
'...\oradata\db02\app01.dbf' , ' ...\oradata\db02\app02.dbf';
```

باستخدام Alter Database:

يمكن نقل ملفات ال Data Files باستخدام القاعدة التالية:

```
ALTER DATABASE  database's name
RENAME FILE 'filename' , ['filename']
TO 'filename' , ['filename'];
```

طريقة التنفيذ:

- اغلاق ال Database.
- نسخ ملف ال Data File الى المواقع الجديدة (القيام بعملية النسخ كما في نظام التشغيل ، أي في الويندوز يستخدم الفأرة مع COPY و PASTE).
- تشغيل ال Database في حالة ال Mount.
- تنفيذ القاعدة السابقة.
- تشغيل ال Database في حالة Open.

شروط الاستخدام:

- **يجب أن تكون ال Database في حالة ال Mount.**
- **يجب تواجد ملفات ال Data Files المراد نقلها في الموقع المحدد في القاعدة.**
- **يمكن نقل Data Files التابعيين لل System Tablespace.**

تذكر: هي نفس القاعدة التي استخدمناها لنقل أو تغيير اسماء ال Redo Log Files.

مثال تطبيقي 5.15:

بعد تنفيذ الثلاث نقاط الأولى من طريقة التنفيذ:

```
ALTER DATABASE app_data RENAME FILE
'...\oradata\db01\app01.dbf'
TO
'...\oradata\db02\app01.dbf';
```

:TABLESPACE WITH OMF

يمكن تكوين Tablespace تتبع نظام ال OMF بتحدد العامل "Parameter" DB_CREATE_FILE_DEST لل Data Files التابعيين لل Tablespace واستخدام قاعدة ال CREATE TABLESPACE مع عدة اختلافات مثل عدم الضرورة لذكر اسم ال Data Files في القاعدة.

```
CREATE TABLESPACE tablespace's name
[DATAFILE [filename] [SIZE number [K|M] ] ;
```

عند تكوين ال Tablespace بنظام ال OMF تكون ال Data Files:

- **حجمها ال Default 100MB (كما في المثال 5.16)**
- **تستخدم الطريقة الأوتوماتيكية لزيادة حجمها (AUTOEXTEND) مع عدم تحديد أقصى حد للحجم (UNLIMITED).**

مثال تطبيقي 5.16:

لتكوين Tablespace بنظام ال OMF يجب تحديد العامل "Parameter" DB_CREATE_FILE_DEST ثم كتابة التالي:

```
CREATE TABLESPACE new_data;
```

:QUERYING INFORMATION

للحصول على معلومات حول ال Tablespaces يمكن استخدام:

- **DBA_TABLESPACES:** يوفر هذا ال View معلومات حول جميع ال Tablespaces في ال Database مثل اسم ال Tablespaces ونوعية ال Tablespace (Temporary, Undo) ونوعية ادارة ال Extents (Locally, Dictionary) وغيرها.

- **V$TABLESPACE:** توفر معلومات عن اسم و رقم ال Tablespaces.

للحصول على معلومات حول ال Data Files يمكن استخدام:

- **DBA_DATA_FILES:** يوفر معلومات حول ال Data Files مثل اسم Data Files وال Tablespace التي ينتمي لها ال Data File وحجم ال Data Files وغيرها من المعلومات.

- **V$DATAFILE:** يوفر معلومات حول ال Data Files مثل حالة ال Data File (Online,Offline) ، حجم ال Data Files وغيرها.

للحصول على معلومات حول ال Temp Files يمكن استخدام:

- **DBA_TEMP_FILES:** يوفر معلومات حول ال Temp Files مثل اسم Temp Files و ال Tablespace التي ينتمي لها ال Temp File وحجم ال Temp Files وغيرها من المعلومات.

- **V$TEMPFILE:** يوفر معلومات حول ال Temp Files مثل حالة ال Temp File (Online,Offline) ، حجم ال Temp Files وغيرها.

الفصل السادس

السيجمنت و خصائص التخزين

SEGMENTS & STORAGE STRUCTURES

SEGMENTS

ال Segment هي احدى مكونات ال Logical Structure وتأتي بعد ال Tablespace في الترتيب ، يمكن لل Tablespace أن تحوي أكثر من Segments كما يمكن لل Segment أن تتكون من أكثر من Extent.

يوجد عدة أنواع من ال Segment مثل:

- TABLE SEGMENT.
- TABLE PARTITION SEGMENT.
- CLUSTER SEGMENT.
- INDEX SEGMENT.
- INDEX-ORGANIZED TABLE SEGMENT.
- INDEX PARTITION SEGMENT.
- TEMPORARY SEGMENT.
- UNDO SEGMENT.
- LOB SEGMENT.
- NESTED TABLE SEGMENT.

سوف يأتي ذكر كل نوع لاحقاً.

DATA BLOCKS

هي أصغر وحدة تخزين في الأوراكل ويتم تحديد حجمها بواسطة العامل "Parameter" DB_BLOCK_SIZE عند تكوين ال Database ولا يمكن تغيير الحجم فيما بعد. يتكون ال Data Block من:

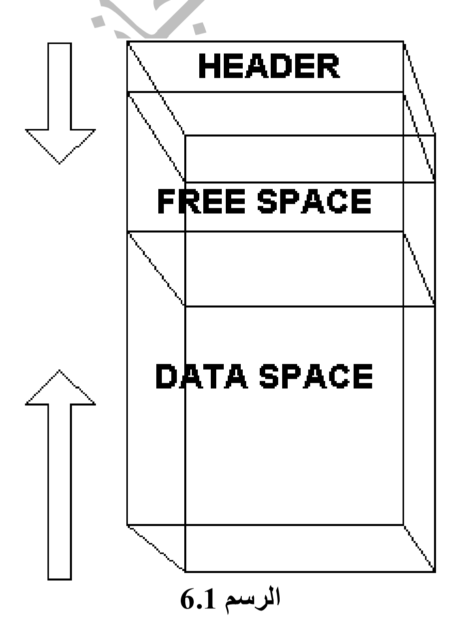
- **BLOCK HEADER**: يحتوي على معلومات حول ال Data Block ونوعه (Data, Index, or Undo). يحتوي أيضاً على ال Table Directory الذي يوفر معلومات حول ال Tables التي لها بيانات (Data Rows) في ال Data Block. ومن أجزاء ال Block Header ايضاً ما يعرف باسم Row Directory الذي يوفر معلومات حول البيانات (Rows) المخزنة في ال Block مثل ال *Row Address*. الجزء الأخير من ال Block Header هي ال Transaction Slots التي تستخدم عندما تحدث تغيرات في البيانات "Rows" نتيجة للمهمات "Transactions" المختلفة.

- **DATA SPACE**: هي المساحة التي يتم تخزين البيانات بها "Rows".

- **FREE SPACE**: هي مساحة خالية تساعد كل من ال Block Header وال Data Space على التوسع في الحجم اذا لزم الأمر. في حال حدوث عمليات كثيرة تسبب أخذ مساحة منها ثم اعادة مساحة لها بشكل متكرر يحدث Fragmentation ويقوم الأوراكل بعملية Coalesce لها.

تذكر: عملية ال Coalescing من درس ال Locally Managed Tablespace.

تذكر2: يتم تخزين رقم ال Log Sequence Number ورقم ال Checkpoint في ال Data Files Header.



الرسم 6.1

ملاحظة: الأسهم في الرسم 6.1 تشير الى الطريقة التي تكبر بها مساحة كل من ال Data Space وال Block Header وهي طريقة عكسية ، من أسفل الى أعلى ومن أعلى الى أسفل على التوالي.

:BLOCK STORAGE PARAMETERS

عندما يتم تكوين Table أو عنصر أخر "Object" يمكن تحديد عوامل تتحكم في ال Data Blocks التي تنتمي لهذا العنصر. تحديد هذه العوامل مهم جداً ، لأنها تعمل على توفير مساحة كبيرة وتحسين الادارة والعمل.

- **PCTFREE: عند تحديدها ، يتم حجز مساحة من حجم ال Block تخصص للزيادة المتوقعة في البيانات "Data Rows" الناتجة من تغيير "Update" البيانات "Data Rows" في ال Block، الحجم ال Default لها هو 10%.**

- **PCTUSED: تمثل الحجم الأدني للبيانات "Data Rows" في ال Block الذي اذا قل حجم البيانات "Data Rows" المخزنة داخله عن حجم ال PCTUSED ، يتم تحويل ال Block الى قائمة ال Freelist، الحجم ال Default هو 40%.**

- **FREELIST: يمكن لل Segment الواحدة أن يكون لها أكثر من Freelist واحدة بتحديد العامل FREELISTS، ال Default هو لكل Segment واحدة ، قائمة Freelist واحدة.**

- **INITRANS: عند تحديدها يتم حجز مساحة لل "Transaction Slots" في ال Block Header لتخزين البيانات حول المهمات التي تجري. فهي تحدد الحد الأدنى من المهمات "Transactions" التي يمكن أن تجرى على ال Block في نفس الوقت. اذا تم تحديدها ب 3 ، فذلك يعني أنه يخصص ثلاث Transaction Slots بحيث يمكن أجراء ثلاث مهمات على الأقل على ال Block في نفس الوقت ، وعند الضرورة يمكن تخصيص مساحة من ال Free Space في ال Block لإضافة Transaction Slots أخرى، ال Default هو 1.**

- **MAXTRANS: عند تحديدها يتم حجز مساحة لل "Transaction Slots" في ال Block Header لتخزين البيانات حول المهمات التي تجري. فهي تحدد الحد الأعلى من المهمات "Transactions" التي يمكن أن تجرى على ال Block في نفس الوقت. اذا تم تحديدها ب 200 ، فذلك يعني أنه يمكن أجراء 200 مهمة على الأكثر على ال Block في نفس الوقت، Default هو 255.**

ملاحظة: يقصد بال FreeList بأنها قائمة "List" تحدد ال Blocks التي يوجد بها مساحة خالية قادرة على استيعاب بيانات جديدة أو ال Free Blocks.

مثال:

في حال تم تحديد ال PCTUSED بنسبة 30% وال PCTFREE بنسبة 20%:

1- يتم ادخال بيانات "Rows" الى ال Block الى أن تصل نسبة البيانات المدخلة الى 80% أو أقل بحيث يجب أن يظل حجم من ال Block فارغ وهو نسبة ال PCTFREE وهي كما ذكرنا 20%، وهكذا يظل حجم فارغ من ال Block في حال تغير بيانات Rows المخزنة في ال Block مثل تحويل قيمة أحد ال COLUMN من حالة NULL الى قيمة فعلية. هذه المساحة الفارغة مخصصة لل Rows الموجودة في حال تغير قيمها ولا يمكن اضافة Rows جديدة لتحل في الحجم الخالي الناتج من تحديد ال PCTFREE.

2- في حال انخفاض بيانات ال Rows المخزنة في ال Block أو تم حذف عدد من ال Rows ، لا يسمح باضافة Rows جديدة في ال Block اذا لم تنخفض بيانات ال Rows في ال Block عن الحد المحدد ب PCTUSED وهو 30%. في حال انخفاض حجم بيانات ال Rows عن 30%، يتم تحديد أن ال Block جاهز لاستقبال Rows جديدة (Freelist).

MANAGING DATA BLOCKS

يمكن إدارة ال Data Blocks بطريقتين احداهما يدوية والأخرى أوتوماتيكية وهما:

MANUAL DATA BLOCK MANAGEMENT:

يتم استخدام Block Storage Parameters في تحديد حجم ال Blocks.

AUTOMATIC SPACE MANAGEMENT:

يتم استخدام Bitmap عوضاً عن Freelist لتحديد ال Used Blocks وال Free Blocks. يتم حفظ ال Bitmap في Blocks مختلفة تسمى Bitmapped Blocks (BMBs). عندما يتم ادخال بيانات "Rows" جديدة ، يتم البحث في ال Bitmap عن Block يحتوي مساحة خالية مناسبة لحجم البيانات الجديدة المدخلة، وعند حدوث أي تعديلات على ال Blocks (مثل زيادة المساحة الخالية عن المساحة المشغولة) يتم تعديل ال Bitmap لتوافق التغيرات التي طرأت.
فوائد استخدام هذه الطريقة عديدة منها سهولة ادارة ال Blocks اذ يتم تحديد ال PCTUSED و ال FREELIST بشكل أوتوماتيكي ، تخصيص المساحة للعناصر "Objects" بشكل أفضل ، وأداء أفضل خلال عمليات ال INSERT المتزامنة.

نقاط مهمة:

1- يمكن استخدام هذه الطريقة مع Locally Managed Tablespaces فقط.

2- تطبق الطريقة على مستوى ال Tablespace، أي يجب أن تكون جميع ال Segments داخل ال Tablespace تعمل بنفس الطريقة.

3- لا يمكن تغيير الطريقة الى الطريقة اليدوية بعد أن تستخدم.

4- يمكن استخدام هذه الطريقة باضافة جملة SEGMENT SPACE MANAGEMENT AUTO الى جملة CREATE TABLESPACE.

5- لا يمكن استخدام الطريقة مع Tablespace تحوي أو قد تحوي على LOBs.

ملاحظة: يتم تخزين بيانات كبيرة بواسطة LOBs مثل الصور والفيديو وغيرها وسوف يتم التطرق لها في فصل لاحق.

مثال تطبيقي 6.1:

```
CREATE TABLESPACE APP_DATA
DATAFILE '...\oradata\db05\appdata04.dbf'
EXTENT MANAGEMENT LOCAL UNIFORM SIZE 512K
SEGMENT SPACE MANAGEMENT AUTO;
```

STORAGE INFORMATION

للحصول على معلومات حول ال Segments يمكن استخدام:

- **DBA_SEGMENTS:** يمكن الحصول على معلومات حول اسم ال Segment ، ال Tablespace التي تحوي ال Segment ، عدد ال Extents في ال Segment وعدد ال Data Blocks ، وغيرها من المعلومات.

للحصول على معلومات حول ال ***Used Extents*** يمكن استخدام:

- **DBA_EXTENTS:** يوفر معلومات مثل رقم ال Extent ورقم ال Block التي ينتمي لها وعدد ال Blocks ضمن ال Extent وغيرها من المعلومات.

للحصول على معلومات حول ال ***Free Extents*** يمكن استخدام:

- **DBA_FREE_SPACE:** يوفر معلومات مثل اسم ال Tablespace التي تحوي ال Extent وال Blocksالتابعة لها وغيرها.

UNDO SEGMENT

تستخدم ال Undo Segment في حفظ نسخة عن البيانات قبل تعديلها بواسطة المهمات "Transactions" التي أجريت عليها، مثلاً لكي يتمكن الأوراكل من استعادة البيانات قبل التعديل في حال طلب المستخدم Rollback عوضاً عن Commit . يتم استخدام ال Undo Segment Header لتخزين بيانات حول المهمات "Transactions" الحالية التي تستخدم ال Undo Segment.

من فوائد استخدام ال Undo Segment:

- لكي يتمكن الأوراكل من استعادة البيانات قبل التعديل في حال طلب المستخدم Rollback عوضاً عن Commit.
- لكي يتمكن الأوراكل من استعادة البيانات الأصلية في حال تعطل المهمة أو حدوث خلل طارئ لل Instance يتطلب عملية ال Recovery.
- للقيام بما يعرف باسم Read Consistency.

ملاحظة: تستخدم المهمات المرتبطة ببعضها "Serial Transaction"، Undo Segment واحدة فقط ويمكن للمهمات المتزامنة استخدام ذات ال Undo Segment.

تذكر: يتم تخزين معلومات حول ال Undo Segment في:

- Initialization Parameter File.
- Alert Log File.
- Control File.

تذكر2: جاء ذكر ال Undo Segment في درس:

- Undo Tablespace.
- Offline or Online.

System Change Number (SCN):

هو رقم مميز "Unique Number" يتم تكوينه عند حوث أمر Commit، ويساعد في عملية ال Read Consistency وعملية ال Recovery. يتم تخزين ال SCN في ال Control Files، Redo Log Files، و Block Headers.

تذكر: يقوم ال Checkpoint بعمل تجديد لرقم ال SCN في ال Control Files.

Read Consistency:

هي إمكانية رؤية أحد المستخدميين البيانات الأصلية قبل التعديل بالرغم من قيام مستخدم أخر بتعديل البيانات في نفس الوقت. لحدوث ال Read Consistency يجب أن يكون المستخدم الأول قد طلب البيانات قبل أن يقوم المستخدم الثاني بتأكيد التعديلات على التغيرات بواسطة Commit ، عندئذ يحضر الأوراكل البيانات المطلوبة للمستخدم الأول من ال Undo Segment.

مثال:

1- يقوم المستخدم الأول بعمل مهمات على بيانات ال Employees Tables تؤدي الى تغير البيانات مثل عملية **UPDATE**.
2- يقوم الأوراكل بتخزين نسخة لبيانات ال Employees Table قبل التعديل في ال Undo Segment.
3- يقوم المستخدم الثاني بطلب استخراج البيانات عبر استخدام Query مثل **Select * From Employees**، وعندها يتم تحديد رقم ال SCN الذي يستخدم لضمان عدم عرض البيانات التي يحدث لها تغيير ولم يحدث لها Commit.
4- البيانات المعروضة على المستخدم الثاني هي البيانات قبل التعديلات التي أجراها ويجريها المستخدم الأول و هي البيانات المخزنة في ال Undo Segment.
5- يقوم المستخدم الأول بانهاء و تأكيد المهمات التي أجراها باستخدام Commit ، فيتم تجديد ال SCN.
6- يقوم المستخدم الثاني بعد قليل بطلب استخراج البيانات مرة أخرى **Select * From Employees**.
7- البيانات المعروضة هذه المرة هي البيانات بعد التعديلات التي أجراها المستخدم الأول .

Snapshot Too Old Error:

عندما لا يستطيع الأوراكل توفير خدمة ال Read Consistency يظهر هذا الخطأ للمستخدم. يمكن أن يظهر الخطأ عندما يطلب المستخدم الثاني Query يكون البيانات المستخرجة طويلة جداً، بحيث قبل أن ينتهي ال Query من العمل يحدث أمران، الأمر الأول هو أن المستخدم الأول طلب Commit على البيانات ، والأمر الثاني أنه تم استخدام ذات ال Undo Segment من قبل مهمة أخرى بحيث تم تغيير البيانات "Overwritten" في ال Undo Segment.
باستخدام العامل "Parameter" UNDO_RETENTION يمكن تحديد كم من الوقت يسمح للبيانات بالبقاء في ال Undo Table pace وبذلك توفير خدمة ال Read Consistency. اذا تم تحديد قيمة العامل ب 900 ، فهذا يعني أنه يسمح للبيانات البقاء لمدة 15 دقيقة. يمكن تحديد العامل في ال Initialization Parameter File ، ويمكن تعديله ديناميكياً بواسطة **ALTER SYSTEM**.

ملاحظة: اذا كان حجم ال Undo Tablespace صغير، يمكن أن لا تظل البيانات للوقت المحدد بالعامل UNDO_RETENTION ، حيث يتم استخدام عمليات حسابية لتبديل البيانات المخزنة.

UNDO SEGMENT TYPES:

- **SYSTEM UNDO SEGMENT:** تتكون ضمن ال System Tablespace عند تكوين ال Database لتوفر خدماتها للعناصر "Objects" الموجودة ضمن ال System Tablespace.
- **NON-SYSTEM UNDO SEGMENTS:** عندما يوجد Tablespaces في ال Database غير ال System Tablespace ، يجب توفر على الأقل واحدة من Non-System Undo Segment في ال Database لتخدم باقي ال Tablespaces.
- **DEFERRED UNDO SEGMENTS:** يمكن تكوينها عند وضع ال Tablespace في حالة Offline ، اذ يمكن أن تستخدم في عمل Rollback للمهمات "Transactions" التي قد تحدث عند تحويل ال Tablespace الى حالة ال Online.

MANAGING UNDO DATA

يوجد نوعان لإدارة ال Undo Data في الأوراكل هما:

- **AUTOMATIC UNDO MANAGEMENT.**
- **MANUAL UNDO MANAGEMENT.**

سوف يتم التطرق فقط الى النوع الأول في الكتاب.

AUTOMATIC UNDO MANAGEMENT:

يتم تخصيص Undo Tablespace واحدة ذات مساحة كبيرة لكل Instance. يتم التحكم في البيانات الموجودة في ال Undo Tablespace بشكل أوتوماتيكي من قبل ال Oracle Server. يتم تسمية أسماء Undo Segment طبقاً للطريقة التالية _SYSSMUn$ مثل _SYSSMU1$.
لكي تعمل طريقة ال Automatic Undo Management يجب توفر Undo Tablespace واحدة في حالة Active. بالرغم أنه يمكن أن تتواجد أكثر من Undo Tablespace في ال Database ولكن لا يمكن أن يكون أكثر من واحدة في حالة Active.
لاستخدام هذه الطريقة يجب تحديد العامل "Parameter" UNDO_MANAGEMENT بالقيمة AUTO في ال Initialization Parameter File. لا يمكن تغيير قيمة العامل بطريقة ديناميكية بعد تشغيل ال Database،القيمة ال Default هي MANUAL.
لتحديد ال Undo Tablespace التي سوف يتم التحكم بها بواسطة Automatic Undo Management يجب تحديد العامل"Parameter" UNDO_TABLESPACE في ال Initialization Parameter File ، إلا أن يكون هنالك Undo Tablespace واحدة في ال Database عندئذ لا يشترط تحديد هذا العامل "Parameter". يمكن تغيير قيمة هذا العامل بطريقة ديناميكية بواسطة ALTER SYSTEM.

مثال تطبيقي 6.2:

لتغيير قيمة العامل UNDO_TABLESPACE بشكل ديناميكي (للتحديد Tablespace أخرى في حالة Active):

```
ALTER SYSTEM SET UNDO_TABLESPACE = UNDOTBSPACE;
```

أو عبر ال Console باتباع الخطوات التالية:

- **ادخل الى ال Console عبر Standalone.**
- **اضغط على اسم ال Database لكي تظهر نافذة تطلب من اسم المستخدم و كلمة السر.**
- **أدخل اسم المستخدم وكلمة السر مع التاكد من اختيار SYSDBA عوضاً عن Normal.**
- **اضغط على ال Instance Manager.**
- **اضغط على Configuration.**
- **اضغط على قائمة Undo.**
- **أختر من قائمة Current Undo Tablespace اسم ال Undo Tablespace.**
- **اضغط على Apply.**

:Snapshot Too Old Error

يمكن أن يحدث هذا الخطأ ايضاً اذا ما تم تغيير ال Undo Tablespace(بواسطة ال ALTER SYSTEM أو عبر ال Console) وكان ما يزال Query يستخرج البيانات من ال Undo Tablespace التي تم تغييرها.

ملاحظات:

1- في حال لم يتم كتابة جملة ال UNDO TABLESPACE في جملةCREATE DATABASE ، وكانت قيمة العامل UNDO_MANAGEMENT تساوي AUTO ، فإن الأوراكل سيرفر يقوم بتكوين Undo Tablespace بشكل أوتوماتيكي بمسمى SYS_UNDOTBS.
2- يمكن حذف Undo Tablespace باستخدام جملة DROP TABLESPACE التي جاء شرحها سابقاً، ولكن لا يمكن حذف ال Undo Tablespace اذا كانت تستخدم بواسطة ال Instance(Active).

تذكر:

1- يمكن تكوين ال Undo Tablespace بطريقتيين ، إما عبر جملة ال CREATE DATABASE أو عبر جملة CREATE UNDO TABLESPACE.
2- جاء ذكر ال Automatic Undo Management في درس ال Undo Tablespace.

مثال تطبيقي 6.3:

يمكن حذف ال Undo Tablespace بواسطة DROP TABLESPACE ولكن يجب على كل المهمات التي تجري في ال Undo Tablespace أن تنهي عملها قبل اتمام عملية الحذف، يمكن معرفة اذا ما كان هنالك أي عمليات تجري على ال Undo Tablespace أو أي Tablespace من خلال كتابة التالي:

```
SELECT Z.NAME, X.STATUS
FROM V$ROLLNAME Z, V$ROLLSTAT X
WHERE Z.NAME IN
 (SELECT SEGMENT_NAME FROM DBA_SEGMENTS
WHERE TABLESPACE_NAME = 'UNDOTBS')
AND Z.USN = X.USN ;
```

اذا كانت قيمة ال STATUS هي PENDING ONLINE فهذا يعني أن ال Tablespace لا يمكن حذفها، أما اذا كانت القيمة هي PENDING OFFLINE فهذا يعني أنه يمكن حذف ال Tablespace.

:Resource Manager

يمكن استخدام ال Resource Manager للتحكم بالنظام بحيث يتم منع المستخدميين أو مجموعة من المستخدميين من استهلاك مساحة كبيرة لل Undo Data بحيث يتم تخصيص مساحة قصوى "Maxsize" لكل مجموعة لاستخدامها في تخزين ال Undo Data. يتم تخصيص مساحة قصوى بتحديد ال *UNDO_POOL* والذي هو في الحالة الإفتراضية "Default" محدد بالقيمة UNLIMITED.

QUERYING UNDO INFORMATION

يمكن استخدام ال Views التالية للحصول على معلومات حول ال Undo Segment:

- **V$UNDOSTAT:** يساعد في تحديد حجم ال Undo Tablespace ، اذ يوفر معلومات حول عمل ال Undo Tablespace وضغط البيانات عليها خلال 10 دقائق.

```
SELECT BEGIN_TIME, END_TIME, UNDOBLKS
FROM V$UNDOSTAT;
```

- **DBA_ROLLBACK_SEGS:** للحصول على معلومات حول جميع ال Undo Segment في ال Database. وحدها فقط توفر بيانات حول ال Offline Segment.

```
SELECT SEGMENT_NAME, TABLESPACE_NAME, OWNER, STATUS
FROM DBA_ROLLBACK_SEGS;
```

- **V$ROLLNAME:** توفر اسماء جميع ال Online Segments.

```
SELECT USN, NAME FROM V$ROLLNAME;
```

- **V$ROLLSTAT:** توفر بيانات حول ال Online Segments مثل حجم ال Segments وعدد ال Extents وغيرها. يمكن عمل Join مع ال V$ROLLNAME بواسطة ال USN.

```
SELECT A.NAME , B.EXTENTS
FROM V$ROLLNAME A , V$ROLLSTAT B
WHERE A.USN = B.USN;
```

الفصل السابع

إدارة التابل و الإندكس والكونيستراتيت

Managing tables & indexes & constraints

STORING DATA

يتم تخزين ال DATA في أوراكل على شكل Rows و Columns في ال Tables. يوجد عدة أنواع من ال Tables مثل:

- **TABLES (أو Regular Tables):** هي ال Table التي يتم معظم العمل عليها والتي يكون التحكم بتوزيع بال Rows فيها من وظائف ال Oracle Sever.

- **PARTITIONED TABLES:** يكون لل Table أكثر من جزء "Partition"، كل جزء عبارة عن Segment، كل Segment يمكن أن تتواجد في Tablespace مختلفة.

- **INDEX-ORGANIZED TABLE:** يجب تخصيص ال Primary Key لل Table ويتم تخزين ال PK وال Table في ذات المساحة ، وذلك عكس ال Regular Table التي اذا تم تكوين PK لها (الإثنان في مساحة مختلفة "Separate Storage").

- **CLUSTERED TABLE:** يمكن أن تتكون من Table واحدة أو مجموعة من ال Tables التي تستخدم مع بعضها بحيث تتشارك نفس ال Data Blocks. يوجد ما يعرف باسم **Cluster Key** الذي يحدد ال Rows التي يجب أن تخزن مع بعضها البعض.

DATA TYPES:

يوفر الأوراكل مجموعة من Data Types لتخزين البيانات ، تتبع ثلاثة أنواع هم **Scalar, Collection, Relationship**.

Scalar Data Types:

بعض أنواع ال Scalar Data Types:

- **CHAR, NCHAR:** توفر مساحة ثابتة"Fixed-Length" لتخزين الأحرف "Characters". المقصود بالمساحة الثابتة أنه عند تخصيص مثلاً 10 Bytes وتم شغل 7 Bytes لتخزين البيانات، لا تعتبر ال 3bytes المتبقية مساحة خالية بل مساحة مشغولة. أقصى قيمة (مساحة) يمكن اختيارها هي 2000 بايت لكل Row مع العلم أن القيمة ال Default هي 1. تختلف NCHAR عن CHAR أن الأولى يمكن أن تخزن Unicode Character التي توفر خدمة تعدد اللغات.
- **VARCHAR2, NVARCHAR2:** توفر مساحة متغيرة "Variable-Length" لتخزين ال Characters. المقصود بالمساحة المتغيرة أنه عند تخصيص مثلاً 20 Bytes ، واحتاجت البيانات المخزنة في ال Column الى 15 bytes فإنه يتم استخدام 15 Bytes فقط من المساحة الخالية لتخزين البيانات ، ولا يتم حجز المساحة الكلية المحددة. أقصى مساحة يمكن اختيارها هي 4000 بايت لكل Row مع العلم أنه لا يوجد قيمة Default (يجب تحديد القيمة). تختلف NVARCHAR2 عن VARCHAR2 أن الأولى يمكن أن تخزن Unicode Character التي توفر خدمة تعدد اللغات.
- **NUMBER:** توفر مساحة متغيرة لتخزين الأرقام. يمكن أن تستوعب المساحة أرقام تتكون من 38 خانة.

- **DATE:** توفر مساحة ثابتة لتخزين التاريخ والوقت. توفر سبع خانات للقرن والسنة واليوم والساعة والدقيقة و الثانية وأقصى تاريخ يمكن أن يخزن هو ال31 من ديسمبر لعام 9999 ، أما أقدم تاريخ يمكن أن يخزن هو يناير من عام 1.4712 قبل الميلاد.
- **TIMESTAMP:** توفر خدمة مماثلة لل DATE عدا أنها توفر مساحة إضافية لتخزين أجزاء الثانية. توفر 9 خانات لتسجيل أجزاء الثانية (0.987654321) والعدد ال Default هو 6 خانات.
- **TIMESTAMP WITH TIME ZONE:** توفر خدمة مماثلة لل TIMESTAMP ، إلا انها توفر مساحة إضافية لعرض الفارق الزمني بين المستخدم والتوقيت العالمي (توقيت جرينتش). يظهر الفارق على شكل عدد من الساعات والدقائق فقط.
- **TIMESTAMP WITH LOCAL TIME ZONE:** توفر خدمة مماثلة لل TIMESTAMP ، إلا انها توفر مساحة إضافية لعرض ال Session Time Zone، الذي يستخدم في إدخال البيانات بين فروع الشركات البعيدة عن بعضها، أي في حال ادخال البيانات من قبل المستخدم الأول الذي بينه وبين المستخدم الثاني زائد 3 ساعات زمنية ، فإن المستخدم الثاني يرى توقيت إدخال البيانات بتوقيته هو (زائد 3 ساعات زمنية) و ليس بتوقيت المستخدم الأول الذي أدخل البيانات. يظهر الفارق على شكل عدد من الساعات والدقائق فقط.
- **ROW:** توفر مساحة متغيرة لتخزين ال Binary Data. أقصى مساحة يمكن اختيارها هي 2000 بايت.
- **LONG RAW:** توفر خدمة مماثلة لل Row ولكن مساحة أكبر 2GB (لا ينصح باستخدامها)
- **LONG:** توفر مساحة كبيرة لتخزين البيانات ، يمكن أن تصل المساحة الى 2GB (لا ينصح باستخدامها، ويفضل استخدام CLOB أو NCLOB). لا يمكن لل Tableأن تحوي أكثر من Long واحدة.
- **CLOB, NCLOB:** توفر مساحة 4GB لتخزين ال Character Data.
- **BLOB:** توفر مساحة 4GB لتخزين Unstructured Data مثل Binary Images و Documents.
- **BFILE:** باستخدامها، تستطيع تخزين بيانات خارج ال Database في ملفات خارجية، يمكن أن يصل حجم الملف الخارجي الى 4GB.
- **ROWID, UROWID:** توفر مساحة 10 bytes لتخزين رقم مميز لكل Row في ال Database ليستطيع الأوراكل تحديد ال Row بسرعة أكبر عند البحث على البيانات. الفرق بين ROWID وUROWID أن الأخيرة توفر ميزة تسجيل رقم مميز ل Rows تتبع No-Oracle Tables(أي من نوع Database اخر غير الأوراكل).

:Collections Data Types

يوجد نوعان فقط هما:

- **VARYING ARRAYS (VARRAY):** عبارة عن مجموعة مرتبة من البيانات التي تتكون من نفس ال Data Type (مثلاً جميع البيانات أرقام).
- **NESTED TABLES:** عبارة عن مجموعة غير مرتبة من ال Rows ، تخزن ال Rows بشكل منفصل عن ال Table بحيث يوجد وصلة "Pointer" من كل Row الى ال Table.

:Relationship Data Types (REFs)

يتم استخدام Pointers لتدل على بيانات مخزنة في Tables مختلفة.

MANAGING TABLES

:CREATE TABLES

لتكوين Table ، يتم استخدام جملة CREATE TABLE التي من المفروض قد تعرفت عليها بالتفصيل خلال دراستك للامتحان الأول "Introduction to SQL". لكي يستطيع المستخدم تكوين Table في ال Schema الخاص به ، يجب أن يكون لديه CREATE TABLE privilege ، أما لتكوين Table في Schemas لمستخدمين أخرين، يجب أن يكون لديه CREATE ANY TABLE privilege.

ينصح عند تكوين ال Tables:

- توزيع كل Table على Tablespace مختلفة عن ال Tablespaces التي تحوي Undo Segments أو Temporary Segments أو Indexes.
- استخدام نظام Locally Managed (أي بدون استخدام جملة ال Storage) لتجنب ال Fragmentation.

مثال تطبيقي 7.1:

لتكوين Locally Managed Table يمكن كتابة:

```
CREATE TABLE HR.TEST
(FIRST VARCHAR2 (10), LAST NUMBER);
```

أو يمكن اضافة التالي:

```
CREATE TABLE HR.TEST
(FIRST VARCHAR2 (10), LAST NUMBER)
TABLESPACE EXAMPLE
PCTFREE 10 PCTUSED 20
MAXTRANS 200 MINTRANS 1;
```

على اعتبار أن ذكر TABLESPACE EXAMPLE، يدل على اسم ال Tablespace التي تحوي ال Table وهي في هذه الحالة Example Tablespace. وفي حال عدم ذكر هذه الجملة يتم اختيار ال Default وهي عادة ما تكون ال System Tablespace. مع العلم أنه تم تكوين ال Table في ال Schema الخاصة بالمستخدم HR.

أو عبر ال Console باتباع الخطوات التالية:

- ادخل الى ال Console عبر Standalone.
- اضغط على اسم ال Database لكي تظهر نافذة تطلب من اسم المستخدم و كلمة السر.
- أدخل اسم المستخدم وكلمة السر مع التاكد من اختيار SYSDBA عوضاً عن Normal.
- اضغط على + بجانب ال Schema Manager.
- اضغط بواسطة الزر اليمين على Table ثم اختر Create من القائمة.
- ادخل المعلومات المطلوبة (مثل اسم ال Table) ثم اضغط على Apply
- يمكن اختيار Create Using Wizard عوضاً عن Create لتسهيل تكوين ال Table.

مثال تطبيقي 7.2:

لتكوين Dictionary Managed Table يمكن كتابة:

```
CREATE TABLE HR.TEST2
(FIRST VARCHAR2 (10), LAST NUMBER)
TABLESPACE EXAMPLE
PCTFREE 10 PCTUSED 20
MAXTRANS 200 INITRANS 1
STORAGE (INITIAL 200K    NEXT 200K
PCTINCREASE 50
MINEXTENTS 1 MAXEXTENTS 5
FREELISTS 1 FREELIST GROUPS 1
BUFFER_POOL DEFAULT) ;
```

في جملة ال Storage :

INITIAL: تحدد حجم أول Extent.
NEXT: عند زيادة حجم البيانات يتطلب ذلك اضافة Extent، حيث تحدد جحم ال Extent الثانية.
PCTINCREASE: تحدد حجم ال Extents التالية (ليست الثالثة فقط ، إنما كل ما يأتي بعد الثانية) باضافة نسبة مئوية. مثلاً في هذه الحالة يصبح حجم ال Extents التالية 300K (200K+ نصف حجم (50%) ال Extent).
MINEXTENT: الحد الأدنى من ال Extents التي يمكن أن تخصص لل Table.
MAXEXTENTS: الحد الأقصى من ال Extents التي يمكن أن تخصص لل Table.
FREELIST GROUP: عدد مجموعات ال Freelist التي يمكن أن تخصص لل Table.
FREELISTS: عدد ال Freelist ضمن المجموعة.
BUFFER_POOL: تطرقنا له سابقا (Default, Keep, Recycle)

ملاحظة: لا يشترط كتابة كل عوامل ال Storage ، اذ يمكن أخذ القيمة ال Default لكل عامل.

تذكر: ينصح باستخدام ال Locally Managed Table.

أو عبر ال Console باتباع الخطوات التالية:

- ادخل الى ال Console عبر Standalone.
- اضغط على اسم ال Database لكي تظهر نافذة تطلب من اسم المستخدم و كلمة السر.
- أدخل اسم المستخدم وكلمة السر مع التاكد من اختيار SYSDBA عوضاً عن Normal.
- اضغط على + ال Schema Manager.
- اضغط بواسطة الزر اليمين على Table ثم اختر Create من القائمة. ثم ادخل المعلومات المطلوبة مثل بيانات ال Initial Size, Next Size, Increase Size by, Freelists, Groups, Buffer pool.
- اضغط على Apply.

:CREATING TEMPORARY TABLES

يمكن تكوين Temporary Table لتخزين بيانات مؤقته تلزم فقط خلال القيام بأحد المهمات "Transaction" أو طوال فترة عمل ال Session، أي في الحالة الأولى يتم حذف البيانات بعد الإنتهاء من المهمة (مثلاً كتابة Commit)، أما في الحالة الثانية يتم حذف البيانات بعد خروج المستخدم من ال Database.

لتكوين Temporary Table يتم استخدام CREATE GLOBAL TEMPORARY TABLE، ولتحديد اذا ما كانت البيانات سوف تحذف بعد المهمة أو بعد انتهاء ال Session يتم إضافة الجمل التالية:

- **ON COMMIT DELETE ROWS: تحذف البيانات بعد انتهاء المهمة.**
- **ON COMMIT PRESERVE ROWS: تحذف البيانات بعد انتهاء ال Session.**

تذكر: ال Session من الفصل الأول.

ملاحظة: يمكن تكوين Views, Indexes على ال Temporary Table كأي Table أخرى.

ملاحظة2: مهمات ال DML (مثل INSERT, DELETE) التي تنفذ على ال Temporary Table لا تؤدي الى تكوين بيانات في Redo Log Files.

مثال تطبيقي 7.3:

لتكوين Temporary Table يتم حذف بياناتها بعد انتهاء ال Session:

```
CREATE GLOBAL TEMPORARY TABLE TEMP_TABLE
(TEMP_NAME VARCHAR2 (25), TEMP_DATE DATE)
ON COMMIT PRESERVE ROWS;
```

:ALTERING TABLES

يمكن القيام بعدة مهمات باستخدام الجملة ALTER TABLE:

:Changing the Block Storage Parameters

يمكن تغيير قيم PCTUSED, PCTFREE, MAXTRANS, ولا يمكن تغيير ال Tablespace التي تنتمي لها ال Table و MINTRANS.

ملاحظة: يمكن تغيير بعض عوامل جملة ال Storage مثل ال Freelists.

مثال تطبيقي 7.4:

```
ALTER TABLE HR.TEST2
PCTUSED 40
PCTFREE 20
MAXTRANS 250;
```

أو عبر ال Console باتباع الخطوات التالية:

- ادخل الى ال Console عبر Standalone.
- اضغط على اسم ال Database لكي تظهر نافذة تطلب من اسم المستخدم و كلمة السر.
- أدخل اسم المستخدم وكلمة السر مع التاكد من اختيار SYSDBA عوضاً عن Normal.
- اضغط على + ال Schema Manager.
- اضغط على + ال Table.
- اختر اسم المستخدم الذي يملك ال Table المراد تغيير خواصها (مثل HR)
- اختر اسم ال Table من النافذة اليمنى و اضغط عليها مرتين (أو بواسطة الزر اليمين للفأرة ، اختر من القائمة "View\Edit Details".
- ادخل الى قائمة Storage ، وعدل البيانات ثم اختر Apply.

Manual Allocating Extents:

يمكن اضافة Extents بشكل يدوي الى ال Table باستخدام القاعدة التالية:

```
ALTER TABLE [schema.]table_name
ALLOCATE EXTENT
 [
SIZE number [K|M]
DATAFILE ‘filename’
]
```

اذا لم تتم كتابة جملة SIZE number [K|M] ، يتم تحديد حجم ال Extent بناءاً على الحجم المحدد ب NEXT ، أما اذا لم تتم كتابة جملة ال DATAFILE ‘filename’ ، فيتم اضافة Extent الى أي Data File ضمن ال Tablespace التي تحوي ال Table.

مثال تطبيقي 7.5:

لإضافة Extent بشكل يدوي يمكن كتابة:

```
ALTER TABLE HR.TEST2
ALLOCATE EXTENT;
```

:Moving Tables

يمكن نقل Table (بشرط أن لا تكون Partitioned Table) من Tablespace الى Tablespace أخرى باستخدام جملة MOVE مع ALTER SYSTEM.
عند عملية نقل ال Table يمكن تغيير ال Storage Parameters كما في المثال 7.6.

ملاحظة: يمكن تنفيذ العملية لتغيير خواص ال Storage Parameters دون نقل ال Table الى Tablespace أخرى (بنقل ال Table من Segment الى أخرى) كما في المثال 7.6.

ملاحظة2: يحب اعادة تعريف ال Indexes الخاصة بال Table التي تم نقلها لتجنب الأخطاء.

مثال تطبيقي 7.6:

```
ALTER TABLE HR.TEST2 MOVE
TABLESPACE APP_DATA
STORAGE (INITIAL 300K)
PCTUSED 30
INITRANS 2;
```

لتغيير Storage Parameters دون نقل ال Table الى Tablespace أخرى:

```
ALTER TABLE HR.TEST2 MOVE
TABLESPACE EXAMPLE
STORAGE (INITIAL 300K)
PCTUSED 30;
```

مع العلم أن موقع ال HR.TEST2 هو ال Example Tablespace قبل المهمة.

:DROPPING TABLES AND COLUMNS

:Truncate a Table

باستخدام أمر TRUNCATE TABLE ، يتم حذف جميع بيانات ال Table واستعادة المساحة المشغولة (زيادة المساحة الخالية) التي كانت تشغلها البيانات. تعتبر TRUNCATE TABLE من أوامر ال DATA DEFINITION LANGUAGE (DDL)، وبالتالي فإنها لا تسبب تكون Undo Data على اعتبار أن أوامر ال DDL لا يمكن أن يحدث لها Rollback (أي أنه لا يمكن استعادة البيانات بعد حذفها).

ملاحظات:

1- استخدام TRUNCATE TABLE يؤدي الى حذف ال Indexes الخاصة بهذه ال Table.
2- لا يمكن استخدامها على Table لها علاقة Foreign Key مع Table أخرى.

مثال تطبيقي 7.7:

يمكن للمستخدم HR كتابة التالي :

TRUNCATE TABLE TEST2;

أما ال DBA أو أي مستخدم أخر يملك الصلاحيات "Privileges" اللازمة، فيجب ذكر ال Schema:

TRUNCATE TABLE HR.TEST2;

Drop a Table:

يمكن حذف Table بالكامل مع بياناتها باستخدام أمر DROP TABLE. يجب إضافة جملة CASCADE CONSTRAINTS اذا كان لل Table علاقة Foreign key مع Table أخرى.

مثال تطبيقي 7.8:

لحذف Table بالكامل من ال Database:

DROP TABLE HR.TEST2;

Dropping a Column:

يمكن حذف Column والبيانات التي يحويها من ال Table باستخدام ALTER TABLE. قد تتطلب عملية حذف Column مدة طولية جداً و Undo Space كبيرة جداً لل Tables التي تحوي بيانات ضخمة جداً. في هذه الحالة، يفضل وضع ال Column ضمن قائمة ال UNUSED ثم حذفه بعد الأنتهاء من وقت الذروة (أي عندما تكون المهمات التي تجري على ال Table قليلة).
وضع ال Column في حالة ال UNUSED ، هو مشابه لعملية حذفه (لكنه في الواقع موجود)، اذ لا يمكن استخراج البيانات من أو مشاهدته عند استخدام أمر DESCRIBE.
استخدام أمر ALTER TABLE لحذف Column ، يسمح فقط بحذف Column واحدة كل مرة، ولكن باستخدام UNUSED ، يمكن وضع أكثر من Column في هذه الحالة ثم حذفها جميعا في نفس الوقت.
يجب إضافة جملة CASCADE CONSTRAINTS اذا كان لل Column علاقة Foreign key مع Column أخر.

ملاحظة: يحب على الأقل أن يبقى Column واحد في ال Table بعد الحذف وإلا لن تنجح عملية الحذف ، ولا يمكن استرجاع ال Column المحذوف بعد حذفه.

مثال تطبيقي 7.9:

لحذف Column بشكل مباشر:

```
ALTER TABLE HR.TEST
DROP COLUMN LAST CASCADE CONSTRAINTS;
```

في حالة ال Column الذي يحوي بيانات كبيرة ، وتوقفت عملية الحذف قبل حذف ال Column يمكن كتابة:

```
ALTER TABLE HR.TEST
DROP COLUMNS CONTINUE;
```

لوضع ال Column في حالة UNUSED:

```
ALTER TABLE HR.EMPLOYEES
SET UNUSED COLUMN PHONE_NUMBER CASCADE CONSTRAINTS;
```

لحذف ال Unused Columns:

```
ALTER TABLE HR.EMPLOYEES
DROP UNUSED COLUMNS;
```

QUERYING TABLE INFORMATION:

لمعرفة ال Unused Columns في ال Table، يستخدم :

- **DBA_UNUSED_COL_TABS: يوفر معلومات حول ال Columns الموضوعة في حالة UNUSED ، و اسم ال Table واسم المستخدم المالك لل Table.**

للحصول على معلومات حول ال Tables يمكن استخدام:

- **DBA_TABLES: يوفر معلومات عديدة مثل اسم ال Table واسم ال Tablespace التي تنتمي لها ال Table و اسم المستخدم الذي يملك ال Table وخواص ال Storage (مثل PCTUSED) و غيرها.**
- **DBA_OBJECTS: يوفر معلومات حول جميع العناصر "Objects" من ضمنها ال Table. من المعلومات المتوفرة اسم العنصر و توقيت تكوينه و الحالة التي هو عليها و غيرها من المعلومات.**

للحصول على معلومات حول ال Columns في ال Table يمكن استخدام:

- **DBA_TAB_COLUMNS: يوفر معلومات عديدة مثل اسم ال Table واسم ال Column وخواص ال Column وغيرها.**

ملاحظة: كما يوجد view باسم DBA_TABLES يوجد ايضاً USER_TABLES وALL_TABLES. تذكر اقسام ال Dictionary Views في الفصل الثالث.

MANAGING INDEXES

يساعد ال Index في تسريع استخراج البيانات من ال Tables كما يخفف من ال Input/Output الناتج عن ال Queries (لأنه يمنع حدوث بحث كامل على ال Table للحصول على البيانات). يوجد خواص مشتركة بين ال Table و ال Index، حيث كما أنه يمكن تكوين Partitioned Table يمكن أيضاً تكوين Partitioned Index، ايضاً يوجد Storage Parameters لل Index مشابه لل Storage Parameters لل Table.
يمكن أن يتم تكوين ال Index ل Column واحد فقط ويسمى في هذه الحالة **Single Index** ، كما يمكن تكوين ال Index ليشمل أكثر من Column (الحد الأقصى 32 Columns) ويسمى في هذه الحالة **Composite Index**. لا يوجد حد لعدد ال Indexes التي يمكن تكوينها على ال Table ، ويتم ادارة ال Index من قبل ال Oracle Server.

من أنواع ال Index:

- **UNIQUE INDEX:** يمكن تكوين هذه النوع لضمان عدم وجود بيانات متشابه في ال Rows ضمن ال Column الذي تم تكوين Index عليه.
- **NON-UNIQUE INDEX:** يسمح هذه النوع بتشابه بيانات ال Rows في ال Column الذي تم تكوين Index عليه.
- **FUNCTION-BASED INDEX:** يمكن تكوين هذا النوع لتسريع استخراج البيانات من ال Columns التي تحتوي على عمليات مختلفة"Functions" مثل استخدام ال SUBSTR أو INSTR أو REPLACE أو غيرها في ال Column.
- **PARTITIONED INDEX:** يتم تكوين أكثر من INDEX في أكثر من Segment في Tablespaces مختلفة. يستخدم لل Tables التي تحوي بيانات كبيرة جداً وغالباً ما يستخدم لل Partitioned Tables.

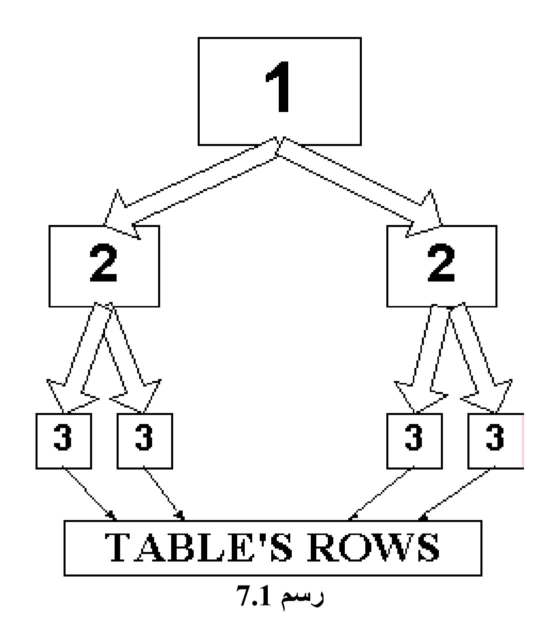
يمكن أن يتم تكوين ال Index اعتماداً على نموذجين هما B-Tree Index أو Bitmap Index.

:B-TREE INDEX

التكوين الداخلى لل Index يكون على شكل شجرة لها أصل وفروع. تتكون الشجرة من ثلاثة أجزاء هي ال Root المتصل بال Branches المتصلة بال Leafs التي تحتوي على ال **Index Entries** التي تشير أو تدل على ال Rows في ال Table باستخدام ال ROWIDs Lists . في الرسم 7.1 يدل الرقم 1 على ال Root ، الرقم 2 على ال Branch والرقم 3 على ال Leaf.

يتكون كل ال Index Entry من أربعة أجزاء هي:

- **HEADER:** يتم تخزين فيه عدد ال Columns التي تعتمد على نوع ال Index (Single, Composite).
- **COLUMN LENGTH:** الذي يحدد حجم ال Column.
- **COLUMN VALUE:** الذي يحدد قيمة ال Column.
- **ROWID:** أرقام ال ROWIDs الخاصة بال Rows في ال Table.



رسم 7.1

ملاحظات:

1- عندما يحدث عملية INSERT لل Table يقوم الأوراكل بإضافة Index Entry جديد ليمثل ال Rows الجديدة المضافة.
2- لا يمكن الإستفادة من B-Tree Index في ال Rows التي تحوي على NULL. أي عند البحث بواسطة جملة WHERE عن قيمة NULL (where last_name = null) فإن الأوراكل يقوم بإجراء بحث كامل على ال Table ولا يقوم باستخدام ال Index.
3- يفضل استخدامها على ال Column الذي يحوي بيانات مختلفة (مميزة) كبيرة مثل أرقام الهاتف وعناوين الموظفين .
4- لا يفضل استخدامها على ال Columns التي تتعرض لجملة ال OR بشكل مستمر.

:BITMAP INDEX

التكوين الداخلى لل Index يشبه تكوين ال B-Tree عدا عن استخدام ROWIDs يتم استخدام خريطة بايتات "Bitmap" لتحديد ال Rows في ال Table بحيث يمثل كل بايت ROWID .

ملاحظات:

1- يفضل استخدامها على ال Column الذي لا يحدث لبياناته تجديد مستمر "Low Update".
2- يفضل استخدامها عند استخدام WHERE أو OR بشكل متكرر.
3- يفضل استخدامها على ال Table التي تحوي بيانات كبيرة جداً ونسبة البيانات المختلفة (المميزة) قليل جداً مثل نوع الإنسان (ذكر أو انثى).
4- عندما يحدث تغيير لبيانات ال Column، يتم تجديد ال Bitmap لتناسب التغيير.

:CREATING B-TREE INDEXES

قبل تكوين ال Index ينصح بالتالي:

- **تكوين أقل عدد ممكن من ال Indexes على Table تجري عليها عمليات DML كثيرة، لأن ال Indexes يبطئ هذا النوع من العمليات.**
- **وضع ال Indexes في Tablespace خاصة.**
- **اختيار NOLOGGING لل Indexes الكبيرة.**
- **قيمة العامل INITRANS يجب أن تكون أكبر من مثيلتها التي حددت على ال Table.**

لتكوين B-Tree Index يجب اتباع القاعدة التالية:

```
CREATE [UNIQUE] INDEX index_name
ON [schema.]TABLE (Column [ASC | DESC] , Column [ASC|DESC] , …)
[TABLESPACE tablespace_name]
[PCTFREE number]
[MAXTRANS number]
[MINTRANS number]
[LOGGING | NOLOGGING]
[NOSORT]
[STORAGE INITIAL number K|M  NEXT number K|M …..]
```

في القاعدة:

- **PCTFREE: هي المساحة التي تحجز في كل Block عند تكوين ال Index لكي تستقبل ال Indexes الجديدة.**
- **NOSORT: تحدد أن ال Rows مرتبة بالترتيب التصاعدي وبالتالي لا يجب على ال Oracle Server ترتيب ال Rows عند تكوين ال Index.**
- **ASC|DESC: تحديد اذا ما كان على ال Oracle Server تكوين ال Index بشكل تصاعدي أو تنازلي.**

ملاحظة: لا يمكن تحديد ال PCTUSED لل Index.

مثال تطبيقي 7.10:

لتكوين B-Tree Non-Unique Index:

```
CREATE INDEX ind_HR_test
ON HR.TEST (FIRST)
PCTFREE 30
STORAGE (INITIAL 200K NEXT 200K PCTINCREASE 0 MAXEXTENTS 50)
TABLESPACE INDX;
```

لتكوين B-Tree Unique Index أضف كلمة UNIQUE بعد CREATE مباشرة.

:CREATING BITMAP INDEXES

لتكوين Bitmap Index يجب إضافة كلمة BITMAP الى جملة CREATE INDEX التي تم استخدامها في المثال 7.10. يمكن تحديد الحجم المخصص من الذاكرة لتخزين الBitmap بتحديد العامل CREATE_BITMAP_AREA_SIZE الذي قيمته في الحالة Default هو 8MB. بزيادة حجم قيمة العامل، يتم تكوين ال Index بسرعة أكبر، ولكن ينصح باستخدام قيمة صغيرة اذا كانت البيانات المميزة (المختلفة) في ال Column قليلة جداً (بيانات مختلفة كبيرة = ذاكرة كبيرة).

مثال تطبيقي 7.11:

```
CREATE BITMAP INDEX ind_HR_jobid
ON HR.JOBS (JOB_ID)
PCTFREE 30
STORAGE (INITIAL 200K NEXT 200K PCTINCREASE 0 MAXEXTENTS 50)
TABLESPACE INDX;
```



مثال تطبيقي 7.12:

يمكن تكوين B-Tree Index أو Bitmap Index عبر ال Console باتباع الخطوات التالية:

- ادخل الى ال Console عبر Standalone.
- اضغط على اسم ال Database لكي تظهر نافذة تطلب من اسم المستخدم و كلمة السر.
- أدخل اسم المستخدم وكلمة السر مع التاكد من اختيار SYSDBA عوضاً عن Normal.
- اضغط على + ال Schema Manager.
- اضغط على مجلد Index ثم بواسطة الزر اليمين للفأرة اختر من القائمة Create.
- ادخل اسم ال Index وأختر ال Schema و ال Tablespace من القوائم ، ثم اختر ال Column أو ال Columns المراد وضع Index لها من قائمة Table Columns.
- لتكوين Bitmap Index ، ضع علامة داخل المربع الصغير (فوق زر Create) بجانب كلمة Bitmap.
- أختر باقي البيانات من القوائم الأخرى (Storage, Partitions, Options)
- اضغط على زر Create.

:ALTERING INDEXES

يمكن القيام بعدة مهمات باستخدام جملة **ALTER INDEX**:

:Changing the Block Storage Parameters

يمكن تغيير قيم بعض العوامل مثل MAXTRANS ويمكن تغيير بعض عوامل جملة ال Storage مثل ال MAXEXTENTS.

مثال تطبيقي 7.13:

```
ALTER INDEX ind_HR_jobid
MAXTRANS 250
STORAGE (MAXEXTENTS 200);
```

أو عبر ال Console باتباع الخطوات التالية:

- **ادخل الى ال Console عبر Standalone.**
- **اضغط على اسم ال Database لكي تظهر نافذة تطلب من اسم المستخدم و كلمة السر.**
- **أدخل اسم المستخدم وكلمة السر مع التاكد من اختيار SYSDBA عوضاً عن Normal.**
- **اضغط على + ال Schema Manager.**
- **اضغط على + ال Index.**
- **اختر اسم المستخدم الذي يملك ال Index المراد تغيير خواصها (مثل HR)**
- **اختر اسم ال Index من النافذة اليمنى و اضغط عليها مرتين (أو بواسطة الزر اليمين للفأرة ، اختر من القائمة "View\Edit Details".**
- **ادخل الى قائمة Storage ، وعدل البيانات ثم اختر Apply.**

:Allocating & Deallocating Extents

يمكن اضافة Extents الى مساحة ال Index بنفس الطريقة المستخدمة في إضافة Extent الى ال Table بنفس الشروط. كما يمكن حذف Extents فارغة غير مستخدمة من مساحة ال Index بتعويض جملة ال ALLOCATE EXTENT بالجملة التالية:

DEALLOCATE UNUSED [KEEP number [K|M]]

في القاعدة:

- **KEEP: لإبقاء مساحة خالية من ال Unused Extents.**

مثال تطبيقي 7.14:

لإضافة Extent :

```
ALTER INDEX ind_HR_jobid
ALLOCATE EXTENT SIZE 200K;
```

لحذف مساحة خالية (Extents):

```
ALTER INDEX ind_HR_jobid
DEALLOCATE UNUSED;
```

Rebuilding Indexes:

لنقل ال Index الى Tablespace أخرى أو لتغيير خواص ال Storage Parameters يمكن استخدام ALTER INDEX مع REBUILD كما توضح القاعدة:

```
ALTER INDEX [schema.]Index_name REBUILD
[TABLESPACE tablespace_name]
[PCTFREE number]
[INITRANS number]
[MAXTRANS number]
[LOGGING | NOLOGGING]
[REVERSE | NOREVERSE]
[STORAGE (INITIAL SIZE [K|M]  NEXT SIZE [K|M] ....]
```


في القاعدة:

- **REVERSE: يمكن تحويل ال Index الى نوع Reverse Key Index باضافة الجملة الى القاعدة. هذا النوع يعرف بيانات ال Column بشكل عكسي ، مثلاً لو كان Order_ID يحتوي على الرقم 12345 فإن الأوراكل يعكس الرقم الى 54321 ويضيف ال Column الى ال Index. هذا النوع يستخدم بالعادة مع ال Columns المعرفة بياناتها بشكل تصاعدي، اذ يتم توزيع البيانات المتقاربة الى Leafs مختلفة ليتم تسهيل أخراج البيانات بسرعة أكبر.**

نتائج عملية ال Rebuild Index :

- **بعد انتهاء العلمية، يتم تكوين Index جديد و حذف الأصلي، بحيث تساعد على تكوين Index جديد متكامل غير ال Index الأصلي الذي قد يكون قد فقد جزء من مساحة نتيجة للتغيرات التي تطرأ على ال Rows مثل حذف البيانات و تجديدها (Fragmentation) ، وبالتالي استخدام وتخصيص المساحة بشكل أفضل.**

- **خلال العملية ، يمكن لل Queries استخدام ال Index الأصلي.**

ملاحظة: لا يمكن استخدام Rebuild لتغيير ال Index من B-Tree الى Bitmap أو بالعكس.

مثال تطبيقي 7.15:

للقيام بعملية Rebuild على ال Index:

```
ALTER INDEX ind_HR_Jobid REBUILD
TABLESPACE INDX02;
```

:Rebuilding ONLINE

أحياناً تستهلك عملية Rebuild وقت كبير جداً عندما تكون ال Table كبيرة جداً. في السابق لم يكن بالمقدور القيام بمهمات ال DML على ال Table التي يجري عملية Rebuild Index على Index مخصصة لها. في النسخ الحديثة من أوراكل (Oracle8i و Oracle9i) يمكن أن تجري مهمات ال DML بينما تتم عملية Rebuild Index بإضافة ONLINE الى جانب REBUILD، ولكن لا ينصح بالقيام بمهمات كبيرة ويفضل الاقتصار على المهمات الصغيرة.

ملاحظة: لا يسمح بكل الأحوال القيام بمهمات ال DDL على ال Table خلال ال Rebuild Index.

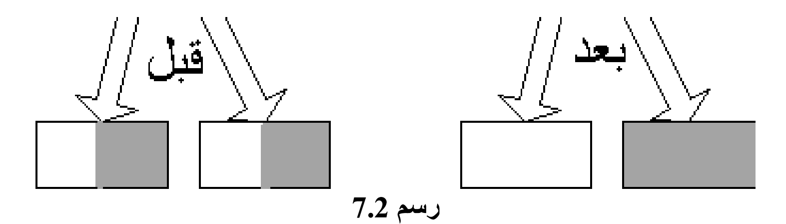
مثال تطبيقي 7.16:

للقيام بعملية Rebuild Online على ال Index:

```
ALTER INDEX ind_HR_Jobid REBUILD ONLINE;
```


:Coalescing Indexes

يمكن التخلص من أثار ال Fragmentation في ال Index بالقيام بعملية Coalesce على ال Index. كما يبدو في الرسم 7.2، وجود 2 leafs تعرضتا لعمليات أدت الى حدوث Fragmentation فأصبحت كل واحد منهما نصف ممتلئة، ولكن بعد حدوث Coalesce تم دمج النصفين مع بعضهما مما نتج عن وجود Leaf فارغة تماما.



رسم 7.2

مثال تطبيقي 7.17:

للقيام بعملية Coalescing لل Index:

```
ALTER INDEX HR.ind_hr_jobid COALESCE;
```

<u>Identify Unused Indexes:</u>

يمكن وضع Index تحت المراقبة لمعرفة اذا كان يستخدم أم لا، وبالتالي حذفه في حال عدم استخدامه. يتم جمع المعلومات حول ال index المراقب في V$OBJECT_USAGE.

ملاحظة: كل مرة يتم فيها مراقبة ال Index ، يتم إلغاء البيانات السابقة لل Index في ال V$OBJECT_USAGE.

مثال تطبيقي 7.18:

لوضع ال Index تحت المراقبة:

```
ALTER INDEX HR.ind_hr_jobid MONITORING USAGE;
```

لوقف مراقبة ال Index:

```
ALTER INDEX HR.ind_hr_jobid NOMONITORING USAGE;
```


<u>ANALYZING INDEXES:</u>

يمكن استخدام جملة ANALYZE INDEX للقيام:

- **التأكد من أن ال Blocks الخاصة بال Index لا تحتوي على بيانات فاسدة " Block Corruption".**
- **للحصول على معلومات حول ال Index.**

بعد القيام ب ANALYZE INDEX يتم تخزين معلومات حول ال Index في View اسمه INDEX_STATS ، يعرض هذا ال View معلومات عن عدد ال Blocks وعدد ال Blocks المستخدم ولكن أهم من ذلك هو رقمي كل من LF_ROWS و DEL_LF_ROWS ، بحيث اذا زادت النسبة بين الرقمين عن 30% (مثلاً الرقم الأول يساوي 50 والثاني يساوي 100، فهذا يعني النسبة 50%) فهذا يعني أن ال Index تعرض لحذف بيانات "Rows" كثيرة مما قد ينتجه عن Fragmentation في ال Index.

مثال تطبيقي 7.19:

للقيام بعملية Analyze لل Index:

```
ANALYZE INDEX HR.JOB_ID_PK VALIDATE STRUCTURE;
```

بعد ذلك استخرج المعلومات من INDEX_STATS:

```
SELECT BLOCKS , PCT_USED, LF_ROWS , DEL_LF_ROWS
FROM INDEX_STATS;
```

يظهر في جهازي الناتج التالي:

```
BLOCKS   PCT_USED   LF_ROWS DEL_LF_ROWS
----------- ---------- ---------- ----------
      0         19          9         16
```

وكما تلاحظ النسبة بين الرقمين لا تتجاوز ال 30%.

DROPPING INDEXES:

عند عدم الحاجة لل Index أو عند حدوث خلل له أو عند تحول حالة ال Index الى INVALID بسبب حدوث خلل لل Instance ، يمكن حذف ال Index بواسطة جملة DROP INDEX.

ملاحظة: لا يمكن حذف ال Index اذا كان Primary Key أو Unique Key ولديه علاقة مع FK.

مثال تطبيقي 7.20:

لحذف Index:

```
DROP INDEX hr.ind_hr_jobid;
```

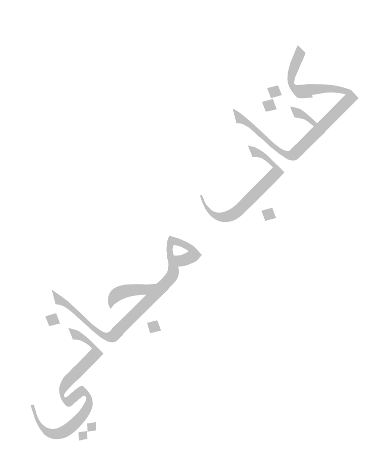
أو عبر ال Console باتباع الخطوات التالية:

- ادخل الى ال Console عبر Standalone.
- اضغط على اسم ال Database لكي تظهر نافذة تطلب من اسم المستخدم و كلمة السر.
- أدخل اسم المستخدم وكلمة السر مع التاكد من اختيار SYSDBA عوضاً عن Normal.
- اضغط على + ال Schema Manager.
- اضغط على + ال Index.
- اختر اسم المستخدم الذي يملك ال Index المراد حذفه (مثل HR)
- اختر اسم ال Index من النافذة اليمنى و بواسطة الزر اليمين للفأرة ، اختر من القائمة "Remove" (أو من القائمة الأساسية، Object ثم Remove).
- اختر Yes.

:QUERYING INDEX INFORMATION

للحصول على معلومات حول ال Index يمكن استخدام:

- **DBA_INDEXES: يوفر معلومات عديدة مثل اسم ال Index ونوعه واسم ال Tableالتي ينتمي لها ال Index واسم المستخدم الذي يملك ال Index وخواص ال Storage وغيرها.**
- **DBA_IND_COLUMNS: يوفر معلومات عديدة مثل اسم ال Index واسم ال Column التي وضع عليها ال Index وخواص ال Column وغيرها.**
- **DBA_IND_EXPRESSIONS: يوفر معلومات حول ال Function-Based Indexes.**
- **V$OBJECT_USAGE: يوفر معلومات ناتجة عن عملية Monitoring Index. يوفر معلومات حول مثل اسم ال Index ونوعه واسم ال Tableالتي ينتمي لها ال Index واذا ما كان يحرى عملية Monitoring على ال Index أم لا (USED التي تدل على اذا ما كان تم استخدام ال Index خلال المراقبة)، وقت بداية وانتهاء المراقبة "Monitoring".**



MANAGING CONSTRAINTS

يمكن وضع قوانين على ال Data (مثلاً عدم السماح بوجود بيانات متشابه في Column)، باستخدام كل من Application Code ، Database Triggers ، و Integrity Constraints.
ال Application Code عبارة عن برمجة كود محدد (مثل برمجة الجافا) يعمل كبرنامج لوحده في ال Client أو من ضمن ال Database على شكل Procedure. أما ال Database Triggers فهي برامج من تكوين PL/SQL ، يتم تنفيذها عند حدوث مهمة محددة في ال Database مثل القيام ب INSERT أو UPDATE.

:INTEGRITY CONSTRAINTS

يفضل استخدام هذا النوع عن النوعيين الأخريين لسهولة تكوينه وتعديله ، ولسهولة تشغيله أو وقف تشغيله، ولإمكانية استخراج معلومات عنه من ال Data Dictionary. يوجد عدة أنواع من ال Integrity Constraint هي:

- **NOT NULL:** يحدد أنه لا يمكن لل Column أن يحتوي على NULL(كل ال Rows يجب أن تحتوي على بيانات ، لا يمكن أن يكون أحدها فارغاً)
- **UNIQUE:** تحدد أن القيم الموجودة في ال Column(أو أكثر من Column) مميزة ولا يمكن أن تتشابه، يستخدم هذا النوع عادة عند تخصيص رقم مميز لكل موظف في الشركة. يسمح ال UNIQUE باستخدام قيم NULL في ال Rows.
- **PRIMARY KEY:** تحدد أن ال Column (أو أكثر من Column) هو ال Primary Key في ال Table. تكون مهمة ال PK بضمان عدم تشابه بيانات ال Rows في ال Column وعدم إحتواء ال Rows على قيمة NULL.
- **FOREIGN KEY:** تحدد أن ال Column (أو أكثر من Column) هو Foreign Key في ال Table ، يتصل مع PK أو Unique في ال Table أو في Table أخرى.
- **CHECK:** تحدد قانون يجب على كل ال Rows تنفيذه (مثل أن يكون قيم ال Rows الخاصة بجنس الإنسان، إما M أو F ، ولا يمكن ادخال أي بيانات أخرى).

حالات ال Integrity Constraints:

يمكن وضع ال Constraints في أربع حالات تؤثر على البيانات المخزنة في ال Table والبيانات الجديدة التي سوف يتم ادخالها مستقبلاً:

- **DISABLE NOVALIDATE:** في حال وضع ال Constraint في هذه الحالة، لا يتم فحص البيانات الموجودة ولا يتم فحص البيانات المستقبلية لمعرفة اذا كانت تتطابق مع القانون أو الشرط الذي يحدده ال Constraint.
- **DISABLE VALIDATE:** يتم ايقاف عمل ال Constraint وحذف ال Index المخصص لها وعدم السماح بالقيام بأي تعديلات على البيانات المخزنة (عدم السماح بعمليات DML).
- **ENABLE NOVALIDATE:** لا يتم فحص البيانات المخزنة لمعرفة اذا ما كانت تتطابق مع قانون ال Constraint و لكن لا يسمح بادخال بيانات جديدة تتعارض مع ال Constraint.
- **ENABLE VALIDATE:** يتم فحص البيانات الجديدة والبيانات المستقبلية والتأكد أنها تتطابق مع ال Constraint. في حال وجود بيانات لا تتطابق مع ال Constraint فيجب حذفها أو تغييرها وإلا لا يمكن الدخول الى هذه الحالة.

ملاحظات:

1- لا ينصح التحويل من الحالة الأولى أو الثانية مباشرة الى الحالة الرابعة ، اذ أن ذلك يؤدي الى عمل Locks على كل ال Table مما قد يؤدي الى تأخر انجاز مهمات ال DML، لذلك ينصح بالتحويل الى حالة Enable Novalidate أولاً قبل التحويل ال Enable Validate.

2- عندما يتم تحويل Constraint تستخدم Primary Key أو Unique Key من حالة Enable الى حالة Disable ، فإن ال Index الموجود يتم حذفه والعكس صحيح.

3- التحول من حالة Enable Novalidate الى حالة Enable Validate ، لا يوقف عمليات ال DDL.

:CONSTRAINTS CHECKING

يمكن لل Constraints أن يقوم بفحص البيانات المدخلة الى ال Table إما عند نهاية كل مهمة من مهمات ال DML و يسمى في هذه الحالة Nondeferred أو Immediate ، أو يمكن لل Constraints أن يقوم بفحص البيانات عند كتابة Commit (قد يشمل مجموعة من المهمات) ويسمى في هذه الحالة Deferred، في كلتا الحالتين يتم عمل Rollback للمهمات التي انجزت اذا لم تتوافق مع قانون ال Constraint.
يفضل استخدام الحالة الثانية التي تنقسم الى قسمين بحيث تستطيع أن تشابه الحالة Nondeferred والحالة Deferred معاً، القسمين هما:

- INITIALLY IMMEDIATE: تعمل كعمل ال Nondeferred.
- INITIALLY DEFERRED: تعمل كمل ال Deferred.

لوضع ال Constraint في احدى الحالتين يتم استخدام القاعدة التالية:

```
SET CONSTRAINT | CONSTRAINTS
(constraint_name | ALL)
(IMMEDIATE | DEFERRED)
```

يمكن أيضاً استخدام ALTER SESSION كما في القاعدة (تعمل في ال Session المحدد فقط):

```
ALTER SESSION
SET CONSTRAINT[S] = (IMMEDIATE | DEFERRED | DEFAULT)
```

مثال تطبيقي 7.21:

لتغيير حالة كل ال Constraints الى Deferred:

```
SET CONSTRAINTS ALL DEFERRED;
```

:CREATING CONSTRAINTS

يمكن تكوين ال Constraints باستخدام جملة **CREATE TABLE** أو جملة **ALTER TABLE**.
يمكن تكوين ال Constraints في ال **Column-Level** أو في ال **Table-Level** على أن الأخيرة تسمح بتكوين ال Constraint لأكثر من Column واحد وهذا لا تسمح به الأولى.
يجب أن يكون للمستخدم **REFERENCES Privilege** اللازمة لكي يستطيع تكوين Constraints في Schemas أخرى غير ال Schema الخاص به.

لتكوين Constraints في ال Column-Level:

```
CREATE TABLE table_name
(column_name TYPE [CONSTRAINT constraint_name] C1 | C2 | C3 | C4| C5)
[NOT DEFERRABLE | DEFERRABLE [INITIALLY (IMMEDIATE |
DEFERRED)] ]
[DISABLE | ENABLE [VALIDATE | NOVALIDATE]
```

أما ال C1, C2, C3, C4 فهي أنواع ال Constraints:

C1= [NOT] NULL
C2= UNIQUE [USING INDEX ….]
C3= PRIMARY KEY [USING INDEX…]
C4= REFERENCES [schema.] table_name [(column_name)] [ON DELETE CASCADE]
C5= CHECK (condition)

اما جملة **USING INDEX** فهي تتطابق مع جملة **CREATE INDEX** عدا استخدام **USING** عوضاً عن **CREATE** وعدم تحديد الجزء **ON [schema.]TABLE (Column [ASC | DESC]**، وعدم ذكر اسم ال Index.

في القاعدة:

- **ON DELETE CASCADE:** تضمن حذف البيانات المرتبطة في ال Child بواسطة FK مع ال Parent (راجع الأمتحان الأول Introduction to SQL).

مثال تطبيقي 2.22:

لتكوين Constraints في ال Column-Level:

```
CREATE TABLE hr.new_emp (
Emp_id NUMBER(8) CONSTRAINT new_emp_id_pk PRIMARY KEY
DEFERRABLE INITIALLY DEFERRED
USING INDEX STORAGE (INITIAL 100K NEXT 100K) TABLESPACE indx,
Emp_Name VARCHAR2(20) CONSTRAINT new_emp_name NOT NULL,
Salary NUMBER(10) CONSTRAINT new_emp_ck CHECK (salary >0)
);
```

لتكوين Constraints في ال Table-Level:

```
CREATE TABLE table_name
(column_name TYPE , column_name TYPE, …. ,

[CONSTRAINT constraint_name]
PRIMARY KEY (column_name , column_name, …)  [USING INDEX ….]
| UNIQUE (column_name , column_name, …)  [USING INDEX ….]
| REFERENCES [schema.] table_name [(column_name, column_name, …)]
[ON DELETE CASCADE]
| CHECK (condition)

[NOT DEFERRABLE | DEFERRABLE [INITIALLY (IMMEDIATE |
DEFERRED)] ]
[DISABLE | ENABLE [VALIDATE | NOVALIDATE]
);
```

ملاحظة: لا يمكن تكوين NOT NULL Constraint إلا في ال Column-Level.

مثال تطبيقي 2.23:

لتكوين Constraints في ال Table-Level:

```
CREATE TABLE hr.new_emp
 (
Emp_id NUMBER(8)
Dept_id NUMBER (8)
Emp_Name VARCHAR2(20) CONSTRAINT new_emp_name NOT NULL,
Dept_name VARCHAR2(20),
Salary NUMBER(10),
CONSTRAINT new_emp_id_pk PRIMARY KEY (Emp_id , Dept_id)
DEFERRABLE
USING INDEX STORAGE (INITIAL 200K NEXT 200K) TABLESPACE indx
CONSTRAINT new_emp_ck CHECK (salary > 0)
);
```

ملاحظة: لتكوين Constraint عبر ال Console اتبع خطوات تكوين ال Table مع التأكد من ادخال البيانات في القائمة Constraints.

تذكر: يفضل توزيع ال Indexes الى Tablespace مختلفة عن ال Table.

:USING ALTER TABLE

يمكن استخدام جملة **ALTER TABLE** لتكوين أو تعديل حالة ال Constraints.

:Creating Constraints

كما ذكرنا قبل قليل أنه يمكن تكوين Constraints باستخدام **CREATE TABLE** أو باستخدام جملة **ALTER TABLE** كما في المثال 2.24.

مثال تطبيقي 2.24:

لإضافة Constraints بواسطة **ALTER TABLE**:

```
ALTER TABLE hr.new_emp ADD
 (
CONSTRAINT new_emp_id_fk FOREIGN KEY (Dept_id)
REFERENCES hr.departments (id)
DEFERRABLE
);
```

لا يمكن تطبيق هذا المثال على ال NOT NULL ، اذ يستخدم التالي لإضافة Constraint من هذا النوع، أو للتحويل بين NOT NULL و NULL:

```
ALTER TABLE hr.new_emp MODIFY Dept_name NOT NULL;
```

:Enabling Constraints

يمكن تحويل حالة ال Constraint من Disable الى Enable باستخدام **ALTER TABLE**.

:*Enable Novalidate*

يفضل التحويل الى هذه الحالة أولاً قبل التحويل الى الحالة Validate لل Unique Constraints و ال PK Constraints لأن ذلك يؤدي الى عمل Locks على كل ال Table مما قد يؤدي الى تأخر انجاز مهمات ال DML، مما يؤدي الى تأخر عملية التحويل. تستخدم القاعدة التالية للتحويل من حالة Disable الى حالة Enable Novalidate.

```
ALTER TABLE [schema.] table_name
ENABLE NOVALIDATE {CONSTRAINT constraint_name | PRIMARY KEY |
UNIQUE (column , column2, ….) }
[USING INDEX…..]
```

بحيث يتم اضافة جملة **USING INDEX** في الحلات التي يتم فيها تحويل حالة ال PK Constraint أو ال Unique Constraint فقط ويحب أن تكون الحالة التي تم تكوينهم في البداية Deferrable ويجب أن يكون ال Index الذي تم تكوينه في البداية إما Disabled أو تم حذفه.

:Enable Validate

هي الحالة الإفتراضية "Default" التي يتم فيها التحويل من ال Disable الى ال Enable. بعد عملية التحويل يتم فيها وضع Locks على البيانات حتى تنتهي عملية فحص البيانات الموجودة، وفي حالة عدم وجود Indexes لل PK,UNIQUE Constraints ، يقوم ال Oracle Server بتكوين Index من نوع Non-Unique اذا كان قد تم تكوين ال Constraints في البداية بالحالة Deferrable والعكس صحيح اذ يقوم ال Oracle Server بتكوين Index من نوع Unique اذا كان قد تم تكوين ال Constraints في البداية بالحالة Nondeferrable.

تستخدم ذات القاعدة التي تستخدم للتحويل من حالة Disable الى حالة Enable Novalidate مع استبدال كلمة NOVALIDATE بكلمة VALIDATE (Validate هي ال Default) وإمكانية إضافة جملة EXCEPTIONS التي سوف نتطرق لها بعد قليل.

مثال تطبيقي 2.25:

لتحويل الحالة الى Novalidate:

```
ALTER TABLE hr.new_emp
ENABLE NOVALIDATE CONSTRAINT new_emp_id_pk;
```

لتحويل الحالة الى Validate:

```
ALTER TABLE hr.new_emp
ENABLE VALIDATE CONSTRAINT new_emp_id_pk;
```

لتحويل الحالة باستخدام ال Console:

- ادخل الى ال Console عبر Standalone.
- اضغط على اسم ال Database لكي تظهر نافذة تطلب من اسم المستخدم و كلمة السر.
- أدخل اسم المستخدم وكلمة السر مع التاكد من اختيار SYSDBA عوضاً عن Normal.
- اضغط على + ال Schema Manager.
- اضغط على + ال Table.
- اختر اسم المستخدم الذي يملك ال Table المراد تغيير خواصها (مثل HR)
- اختر اسم ال Table من النافذة اليمنى و اضغط عليها مرتين (أو بواسطة الزر اليمين للفأرة ، اختر من القائمة "View\Edit Details".
- ادخل الى قائمة Constraints ، ثم عدل البيانات ثم اختر Apply.

:USING EXCEPTIONS

تستخدم جملة ال EXCEPTIONS لمعرفة البيانات التي تخالف ال Constraints بعرض معلومات حول ال البيانات "Rows" المخالفة لل Constraint في Table تسمى EXCEPTIONS Table.

للقيام بفحص البيانات قبل القيام بتحويل ال Constraint من حالة Disable الى Enable يمكن اتباع الخطوات التالية:

- تكوين ال Exceptions Table ، اذا لم يتم تكوينها من قبل بتشغيل Script هو utlexcpt.sql من داخل ال SQLPLUS:

```
@ C:\oracle\ora90\rdbms\admin\utlexcpt.sql
```

- تحويل ال Constraint الى Enable Validate بإضافة الجملة التالية الى ALTER TABLE:

```
EXCEPTIONS INTO system.exceptions
```

- في حال ظهور اخطاء تمنع من تحويل حالة ال Constraint الى Enable Validate، استخدم المعلومات التي تكونت في ال Exceptions Table.

```
SELECT * FROM table_name
WHERE ROWID IN (SELECT ROW_ID FROM EXCEPTIONS);
```

- اجراء تعديل للبيانات المخالفة بواسطة UPDATE TABLE.

- تحويل ال Constraint الى Enable Validate بإستخدام ALTER TABLE.

:QUERYING CONSTRAINTS INFORMATION

للحصول على معلومات حول ال Constraints يمكن استخدام:

- DBA_CONSTRAINTS: يوفر معلومات عديدة مثل اسم ال Constraint ونوعه والحالة التي يوجد فيها وغيرها من المعلومات.

- DBA_CONS_COLUMNS: يوفر معلومات عديدة حول ال Columns التي وضع عليها ال Constraints.

الفصل الثامن

إدارة اليوزرز و باسورد سيكيورتي و الريسورسيس

Managing USERS & password SECURITY& RESOURCES

PASSWORD MANAGEMENT

:PROFILES

يستخدم ال Profile للتحكم بال Password وال Resources بواسطة ال DBA عبر تحديد عوامل "Parameters" خاصة بال Profile، من الأمور التي يمكن التحكم بها مثلاً أن يمنع بعض المستخدمين من اجراء مهمات محددة أو إخراج المستخدم من ال Database اذا لم ينفذ مهمات لوقت محدد. عند تكوين ال Database ، يقوم ال Oracle Sever بتكوين ال Default Profile الذي يخصص لجميع المستخدمين ما لم يتم تكوين Profile جديد(كل القيم في ال Default Profile تساوي Unlimited). يمكن أن يخصص "Assign" ال Profile لل Users فقط ولا يمكن تخصيصه لل Roles أو Profiles أخرى، ولا يؤثر تخصيص ال Profile على ال Session الحالية، اذ يجب خروج المستخدم ثم دخوله مرة أخرى الى ال Database لكي تنفذ حدود ال Profile. يتم تخصيص ال Profile لل Users باستخدام جملة CREATE USER أو ALTER USER.

ملاحظة: يمكن اغلاق وتشغيل ال Profile (Enable, Disable).

:MANAGING PASSWORD

بالتحكم بال Password باستخدام ال Profile يمكن:

- تحديد عمر محدد لكلمة السر "Password" يجب على المستخدم تغييرها بعد انتهاء المدة.
- إجبار المستخدم على اختيار كلمة سر معقدة تتبع نظام محددة (مثلاً أن لا تتشابه كلمة السر مع اسم المستخدم)
- اجبار المستخدم عند تغيير كلمة السر عدم اختيار كلمة سر تم استخدامها سابقاً.
- اغلاق ال Account الخاص بالمستخدم عند فشل ادخال كلمة السر لعدد محدد (مثل فشل ادخال كلمة السر لثلاث مرات).

:Locking Account

يمكن غلق ال Account بشكل اوتوماتيكي عند فشل الدخول الى ال Account عند كتابة كلمة سر خاطئة لعدد من المرات، كما يمكن تحديد المدة التي يظل فيها ال Account مغلق عبر تحديد العاملين التاليين:

- FAILED_LOGING_ATTEMPTS: عدد المرات التي يسمح فيها ادخال كلمة سر خاطئة قبل أن يتم غلق ال Account، مثل أن تحدد القيمة بثلاث محاولات وبعد المحاولة الثالثة يغلق ال Account.
- PASSWORD_LOCK_TIME: تحدد قيمته المدة التي يظل فيها ال Account مغلق، وبعد انتهاء المدة يفتح ال Account بشكل أوتوماتيكي (مثلاً بعد 3 أيام).

كما يمكن لل DBA غلق ال Account أو فتحه باستخدام جملة ALTER USER، في حالة غلق ال Account بواسطة ال DBA لا يمكن فتحه أوتوماتيكياً بل يجب أن يتم فتحه عبر ال DBA فقط.

:Password Expiration

يمكن تحديد المدة التي يجب على المستخدمين تغيير كلمة السر "Password" بشكل دوري، كأن يجبر المستخدميين تغيير كلمة السر كل شهر. يمكن ايضاً اعطاء مهلة اضافية عند انتهاء المدة التي يجب تغيير كلمة السر ، بحيث بعد دخول المستخدم الى Database، يظهر رسالة تحذيرية تطلب من المستخدم تغيير كلمة السر خلال فترة محددة وإلا سوف يتم غلق ال Account.

- **PASSWORD_LIFE_TIME:** المدة التي يمكن لكلمة السر أن تظل دون تغيير، في حال عدم تغيير كلمة السر خلال المدة يغلق ال Account وتتحول حالة ال Account الى Expired.
- **PASSWORD_GRACE_TIME:** بعد ان تنتهي المدة التي يمكن لكلمة السر أن تظل دون تغيير، يمكن اعطاء مدة إضافية لتغيير كلمة السر قبل غلق ال Account مع ظهور رسالة تحذيرية لتغيير كلمة السر قبل المدة المحددة.

مثال:

عند تحديد ال PASSWORD_LIFE_TIME بمدة شهر و PASSWORD_GRACE_TIME بمدة ثلاثة أيام يحدث التالي: بعد انتهاء الشهر دون تغيير كلمة السر وكلما دخل المستخدم الى ال Account تظهر رسالة تحذيرية تطالب المستخدم من تغيير كلمة السر خلال 3 أيام وإلا سوف يتم غلق ال Account.

:Password History

يمكن اجبار المستخدم على اختيار كلمة سر لم تستخدم من قبل أبداً أو منعه من استخدام كلمة سر قديمة خلال فترة محددة.

- **PASSWORD_REUSE_TIME:** تحدد عدد من الأيام قبل أن يتم اعادة استخدام كلمة سر قديمة مرة أخرى.
- **PASSWORD_REUSE_MAX:** تحدد عدد المرات التي يمكن ان يتم استخدام كلمة السر القديمة. يمكن اجبار المستخدم بعدم استخدام كلمة سر قديمة أبداً بوضع قيمة العامل صفر.

ملاحظة: لا يمكن تحديد قيمة للعامليين معاً ابداً، اذا تم تحديد قيمة أحد العامليين يجب وضع قيمة الأخر Unlimited.

:Password Verification

يمكن وضع قوانيين لتسمية كلمة السر باستخدام PL/SQL. من القوانيين التي يمكن تحديدها أن لا تكون كلمة السر مطابقة لإسم المستخدم أو يجب أن تحتوي كلمة السر على أرقام و أحرف معاً وكل ما يساعد على تكوين كلمة سر معقدة يصعب التكهن بها.

- **PASSWORD_VERIFY_FUNCTION:** لتحديد ال PL/SQL Function التي سوف تستخدم في وضع قوانيين على اختيار كلمة السر.

ملاحظة: يجب تكوين ال Function في ال SYS Schema ، ويجب ان تكون من نوع Boolean، أي أن الناتج أما صح "True" أو خطأ "False".

يوفر ال Oracle Server ضمن ال utlpwdmg.sql ، VERIFY_FUNCTION التي يمكن ان تستخدم لوضع قوانيين لكلمة السر اذا تم تحديدها في PASSWORD_VERIFY_FUNCTION.

قوانيين ال VERIFY_FUNCTION:

- **أن لا يقل حجم كلمة السر عن اربع خانات، وأن يكون من ضمنها على الأقل حرف واحد ورقم واحد و رمز خاص واحد مثل $.**
- **أن لا تتطابق كلمة السر مع اسم المستخدم.**
- **أن تختلف كلمة السر الجديدة عن التي قبلها على الأقل بثلاث خانات، مثلاً لو كانت كلمة السر القديمة هي N1E$ ، فلا تقبل كلمة السر الجديدة اذا كانت N1Z$3على اعتبار وجود اختلافيين اثنين فقط هما (3,Z).**

CREATING PROFILE TO MANAGE PASSWORD:

يمكن تكوين Profile لإدارة ال Password باتباع القاعدة التالية:

```
CREATE PROFILE profile_name LIMIT
[FAILED_LOGING_ATTEMPTS number]
[PASSWORD_LOCK_TIME number_of_days]
[PASSWORD_LIFE_TIME number_of_days]
[PASSWORD_GRACE_TIME number_of_days]
[{PASSWORD_REUSE_TIME | PASSWORD_REUSE_MAX}
number_of_days]
[PASSWORD_VERIFY_FUNCTION {function_name | NULL | DEFAULT}]
```

في القاعدة:

- **{PASSWORD_REUSE_TIME | PASSWORD_REUSE_MAX}: جمع العامليين بين { } يدل على السماح بوضع قيمة لأحدهما ولا يمكن لكليهما كما تم ذكره مسبقاً.**
- **يمكن وضع القيمة UNLIMITED عوضاً عن (number ،number_of_days).**

ملاحظة: كل العوامل اختيارية واذا لم يتم كتابتها يؤخذ القيمة ال Default وهي Unlimited.

مثال تطبيقي 8.1:

لتكوين Profile للتحكم بكلمة السر :

```
CREATE PROFILE myprofile LIMIT
FAILED_LOGING_ATTEMPTS 3
PASSWORD_LOCK_TIME 5
PASSWORD_LIFE_TIME 15
PASSWORD_GRACE_TIME 7
PASSWORD_REUSE_TIME 15
PASSWORD_VERIFY_FUNCTION verify_function;
```

ملاحظة: لاستخدام الساعات أو الدقائق عوضاً عن الأيام تستخدم القيمة 1/24 للدلالة على يوم واحد ويستخدم 1/1400 للدلالة على دقيقة واحدة.

لتحويل Profile باستخدام ال Console:

- ادخل الى ال Console عبر Standalone.
- اضغط على اسم ال Database لكي تظهر نافذة تطلب من اسم المستخدم و كلمة السر.
- أدخل اسم المستخدم وكلمة السر مع التاكد من اختيار SYSDBA عوضاً عن Normal.
- اضغط على + ال Security Manager.
- اضغط بواسطة الزر اليمين للفأرة على مجلد ال Profile واختر من القائمة التي تظهر Create.
- في النافذة الجديدة، ادخل اسم ال Profile.
- ادخل الى قائمة Password وادخل البيانات التي تريد.
- اضغط على Create.

ملاحظة: سوف يأتي لاحقاً كيفية تخصيص Profile للمستخدم "User".

:ALTERING PROFILE

يمكن تغيير قيم العوامل باستخدام جملة ALTER PROFILE ، كما في المثال 8.2.

مثال تطبيقي 8.2:

```
ALTER PROFILE myprofile
FAILED_LOGING_ATTEMPTS 2
PASSWORD_LOCK_TIME UNLIMITED
PASSWORD_LIFE_TIME 10
PASSWORD_GRACE_TIME 7/24
PASSWORD_REUSE_TIME 10
PASSWORD_VERIFY_FUNCTION verify_function;
```

أو باستخدام ال Console:

- ادخل الى ال Console عبر Standalone.
- اضغط على اسم ال Database لكي تظهر نافذة تطلب من اسم المستخدم وكلمة السر.
- أدخل اسم المستخدم وكلمة السر مع التاكد من اختيار SYSDBA عوضاً عن Normal.
- اضغط على + ال Security Manager.
- اضغط على + ال Profile واختر اسم ال Profile المراد تعديله.
- تظهر الخواص في النافذة اليمنى، عدل البيانات القائمة Password ثم اضغط علىApply .

ملاحظة: التعديلات التي تحدث لل Profile بواسطة ALTER PROFILE أو عبر ال Console لا تؤثر على ال Session الحالي للمستخدمين، بل عند دخول المستخدميين في المرة القادمة (ال Session القادم).

:DROPPING PROFILE

يمكن حذف ال Profile باستخدام جملة DROP PROFILE عبر استخدام القاعدة التالية:

DROP PROFILE profile_name [CASCADE];

في القاعدة:

- **CASCADE: عند كتابتها يقوم الأوراكل بحذف "Revoke" ال Profile من المستخدميين الذين خصص لهم وتحويلهم الى ال Default Profile.**

مثال تطبيقي 8.3:

لحذف Profile:

DROP PROFILE myprofile CASCADE;

أو باستخدام ال Console:

- **ادخل الى ال Console عبر Standalone.**
- **اضغط على اسم ال Database لكي تظهر نافذة تطلب من اسم المستخدم و كلمة السر.**
- **أدخل اسم المستخدم وكلمة السر مع التاكد من اختيار SYSDBA عوضاً عن Normal.**
- **اضغط على + ال Security Manager.**
- **اضغط على + ال Profile واختر اسم ال Profile المراد حذفه ثم بواسطة الزر اليمين للفأرة أختر من القائمة "Remove" (يمكن استخدام قائمة Object ثم Remove عوضاً عن الزر اليميين للفأرة).**
- **اختر Yes.**

ملاحظات:

1- لا يمكن حذف ال Default Profile.
2- عملية الحذف لا توثر على ال Session الحالية للمستخدميين.

RESOURCE MANAGEMENT

بالتحكم بال Resources باستخدام ال Profile يمكن تنفيذ عدد من القيود منها تحديد أقصى وقت يمكن للمستخدم ان يظل متصلاً بال Database ، وأقصى وقت يمكن أن يظل المستخدم في ال Database دون القيام بمهمات "Idle" ، وتحديد عدد ال Sessions التي يمكن للمستخدم أن يحصل عليها في نفس الوقت.
يجب تحديد العامل "Parameter" RESOURCE_LIMIT في Initiliazation Parameter File بالقيمة TRUE لكي نتمكن من التحكم بال Resources ، مع الإعتبار أن القيمة ال Default هي FALSE، يمكن تغيير قيمة العامل بطريقة ديناميكية تظل سارية المفعول الى أن يحدث إغلاق لل Database باستخدام ALTER SYSTEM كما في المثال 8.4.

مثال تطبيقي 8.4:

لتغيير قيمة العامل RESOURCE_LIMIT بطريقة ديناميكية:

```
ALTER SYSTEM SET RESOURCE_LIMIT = TRUE;
```

للتحكم بال Resources يتم تحديد العوامل التالية في ال Profile:

- **CPU_PER_SESSION: يتم تحديد كم من الوقت يمكن ل Session الإستهلاك من وقت ال CPU الذي هو المعالج "Processor" في جهاز الكمبيوتر الذي يتم فيه كل العمليات (يمكن إطلاق عليه اسم "عقل الكمبيوتر")، بحيث يتم تحديد قيمة العامل بأجزاء من الثانية.**
- **CPU_PER_CALL: يتم تحديد كم من الوقت يمكن لجملة واحدة من جمل SQL الإستهلاك من وقت ال CPU، بحيث يتم تحديد قيمة العامل بأجزاء من الثانية.**
- **SESSIONS_PER_USER: تحدد كم Sessions يمكن للمستخدم أن يملك في ذات الوقت. تذكر من الفصل الأول أنه يمكن للمستخدم الواحد اين يكوّن أكثر من Session واحد في نفس الوقت اذا استخدم أكثر من برنامج مثل SQL PLUS و ORACLE FORMS عدا بعض الحالات القليلة.**
- **CONNECT_TIME: يحدد المدة بالدقائق التي يمكن للمستخدم أو ال Session البقاء متصلاً بال Database، بحيث عند انتهاء المدة يتم اخراج المستخدم من ال Database وإنهاء عمل ال Session وعمل Rollback للمهمات التي تعمل أو لم يحدث لها Commit.**
- **IDLE_TIME: يحدد المدة بالدقائق التي يمكن للمستخدم أو ال Session البقاء متصلاً بال Database بدون القيام بمهمات، بحيث عند انتهاء المدة يتم اخراج المستخدم من ال Database.**
- **PRIVATE_SGA: لتحديد المساحة الخاصة المحجوزة لكل Session في ال SGA (خاص فقط ب Shared Server).**
- **LOGICAL_READS_PER_SESSION: تحدد عدد ال Data Blocks التي يمكن قرائتها أو استخراج البيانات منها في ال Session (يمكن أن تمنع استخراج أحجام بيانات كبيرة).**
- **LOGICAL_READS_PER_CALL: تحدد عدد ال Data Blocks استخراج البيانات منها لكل جملة SQL واحدة.**
- **COMPOSITE_LIMIT: هو حاصل جمع كل من CPU_PER_SESSION و CONNECT_TIME وLOGICAL_READS_PER_CALL وPRIVATE_SGA لمعرفة القيمة الكاملة لل Resource Cost.**

:CREATING PROFILE TO MANAGE RESOURCES

يمكن تكوين Profile لإدارة ال Resources باتباع القاعدة التالية:

```
CREATE PROFILE profile_name LIMIT
[SESSIONS_PER_USER n1]
[CPU_PER_SESSION n1]
[CONNECT_TIME n1]
[IDLE_TIME n1]
[CPU_PER_CALL n1]
[LOGICAL_READS_PER_SESSION n1]
[LOGICAL_READS_PER_CALL n1]
[COMPOSITE_LIMIT n1]
[PRIVATE_SGA n2]
```

في القاعدة:

- **n1: هي أحدى هذه القيم UNLIMITED, DEFAULT, NUMBER.**
- **n2: هي أحدى هذه القيم UNLIMITED, DEFAULT, NUMBER K|M.**

مثال تطبيقي 8.5:

لتكوين Profile للتحكم بال Resources :

```
CREATE PROFILE myprofile LIMIT
SESSIONS_PER_USER 3
CPU_PER_SESSION UNLIMITED
IDLE_TIME 15
CONNECT_TIME 90;
```

لتحويل Profile باستخدام ال Console:

- **ادخل الى ال Console عبر Standalone.**
- **اضغط على اسم ال Database لكي تظهر نافذة تطلب من اسم المستخدم و كلمة السر.**
- **أدخل اسم المستخدم وكلمة السر مع التاكد من اختيار SYSDBA عوضاً عن Normal.**
- **اضغط على + ال Security Manager.**
- **اضغط بواسطة الزر اليمين للفأرة على مجلد ال Profile واختر من القائمة التي تظهر Create.**
- **في النافذة الجديدة، ادخل اسم ال Profile.**
- **ادخل البيانات التي تريد في قائمة General.**
- **اضغط على Create.**

:USING DATABASE RESOURCE MANAGER

يمكن استخدام ال Database Resource Manager لتوفير تحكم أكبر لل Oracle Server على ال Resources وكيفية تخصيصها. يجب أن يكون للمستخدم ال Privilege اللازمة للتحكم بال Database Resource Manager وهي ADMINISTER_RESOURCE_MANAGER privilege.

يتوزع ال Database Resource Manager الى أربعة أجزاء هي:

- **CONSUMER GROUP: مجموعة من المستخدميين يتشاركون في ذات قيم ال Resources التي يحتاجونها (كأن يخصص Profile واحد لكل مجموعة لأن لديهم ذات الإحتياجات من ال Resources).**
- **PLAN: خطط لتوزيع ال Resources على ال Consumer Group.**
- **ALLOCATION METHOD: طرق توزيع ال Resources لل Consumer Groups.**
- **PLAN DIRECTIVES: يستخدم من قبل ال DBA لتوزيع ال PLANS على ال Consumer Groups.**

ملاحظة: معلومات أكثر حول ال Database Resource Manager وكيفية استخدامه في دراسة الإمتحان الرابع من امتحانات ال Oracle9i DBA.

تذكر: تطرقنا لل Database Resource Manager في درس Managing Undo Data.

:QUERYING INFORMATION

للحصول على معلومات حول ال Profile:

- **DBA_PROFILES: للحصول على معلومات حول قيم عوامل ال Profile ، يستخدم ال Resource_Type Column لتحديد نوع العوامل المراد استخراجها (أما Password أو Resources وتسمى هنا Kernel).**

```
SELECT * FROM DBA_PROFILES
WHERE PROFILE = 'myprofile'
AND RESOURCE_TYPE = 'KERNEL';
```

لمعرفة كلمة السر للمستخدم (بشكل مشفر) وحالة ال Account:

- **DBA_USERS: معلومات حول اسم المستخدم وكلمة السر وحالة ال Account (مغلق- يعمل).**

للحصول على معلومات حول ال Resource Cost (من استخدام COMPOSITE_LIMIT):

- **RESOURCE_COST: معلومات حول الأربع عوامل التي يحددها Composite_Limit.**

MANAGING USERS

لكي يتمكن ال DBA من منح فرصة الدخول الى ال Database للموظفين ، يجب منح كل واحد منهم Account يتكون من اسم مستخدم وكلمة سر، يعرف هذا ال Account باسم User. يمكن تخصيص لكل مستخدم مميزات و Privileges و Profile ، و Tablespace لتكوين العناصر "Objects" فيها، وTemporary Tablespace لتخزين البيانات المؤقته التي تنتج عن عمليات المستخدم، كما يمكن تحديد مساحة خاصة للمستخدم من كل Tablespace تعرف باسم Tablespace Quotas. كل العناصر "Objects" التي يملكها المستخدم في ال Database (مثل Tables, Views, Indexes) تعرف باسم Schema ولا يمكن للمستخدم الواحد أن يكون له أكثر من Schema واحدة.

:CREATING USERS

لتكوين User جديد تستخدم القاعدة التالية:

```
CREATE USER user_name
IDENTIFIED [BY password | EXTERNALLY]
[DEFAULT TABLESPACE Tablespace_name]
[TEMPORARY TABLESPACE Tablespace_name]
[QUOTA {number K|M | UNLIMITED} ON Tablespace_name, …]
[PASSWORD EXPIRE]
[ACCOUNT {LOCK | UNLOCK}]
[PROFILE {profile_name | DEFAULT}]
```

في القاعدة:

- EXTERNALLY: سوف نتطرق لها بعد قليل.
- DEFAULT TABLESPACE: تحديد ال Tablespace التي سوف تخزن العناصر التي قد يكونها المستخدم مستقبلاً.
- TEMPORARY TABLESPACE: تحديد ال Temporary Tablespace التي سوف يتم تخزين البيانات المؤقتة الناتجة عن مهام المستخدم.
- QUOTA: تحديد مساحة محددة من أي Tablespace تخصص للمستخدم لتخزين عناصر "Objects" فيها.
- PASSWORD EXPIRE: لإجبار المستخدم على تغيير كلمة السر، التي عُرفت بواسطة DBA عند تكوين ال Account، عند دخول المستخدم الى ال Account مباشرة.
- ACCOUNT: لتحديد اذا ما كان يريد ال DBA تكوين ال Account جاهز للعمل مباشرة أو أن يكون مغلق ثم يتم فتحه بعد فترة.
- PROFILE: تحديد ال Profile المخصص للمستخدم.

مثال تطبيقي 8.6:

لتكوين User جديد :

```
CREATE USER Ahmad
IDENTIFIED BY newemp
DEFAULT TABLESPACE data
TEMPORARY TABLESPACE temp
QUOTA UNLIMITED ON users
QUOTA 2M ON indx
QUOTA 10M ON data
PASSWORD EXPIRE;
```

لتكوين User باستخدام ال Console:

- ادخل الى ال Console عبر Standalone.
- اضغط على اسم ال Database لكي تظهر نافذة تطلب من اسم المستخدم و كلمة السر.
- أدخل اسم المستخدم وكلمة السر مع التاكد من اختيار SYSDBA عوضاً عن Normal.
- اضغط على + ال Security Manager.
- اضغط بواسطة الزر اليمين للفأرة على مجلد ال Users واختر من القائمة التي تظهر Create.
- في النافذة الجديدة، ادخل البيانات في قائمة General وقائمة QUOTA.
- اضغط على Create.

Operating System Authentication:

يمكن السماح للمستخدميين الدخول الى ال Database دون استخدام كلمة سر ، هذا يعني بمجرد استطاعة المستخدم الدخول الى نظام التشغيل عبر نظام الحماية المعمول فيه في نظام التشغيل (قد يكون كلمة سر خاصة بنظام التشغيل "Operating System") يستطيع المستخدم الدخول الى ال Database دون الحاجة الى كلمة سر خاصة بالأوراكل. لتطبيق هذا النظام يجب استخدام كلمة EXTERNALLY في جملة CREATE USER. بتحديد العامل OS_AUTHENT_PREFIX يمكن تمييزاسم المستخدم في نظام التشغيل واسم المستخدم في ال Database ، حيث أن القيمة ال Default لهذا العامل هي OPS$.

مثال:

اذا كان المستخدم مؤهل للدخول عبر حماية نظام التشغيل (التي تطلب من المستخدم كتابة اسم و كلمة سر) الى ال Database وكان اسم المستخدم في نظام حماية نظام التشغيل هو Ahmad فإن اسم المستخدم في ال Database يكون OPS$Ahmad.

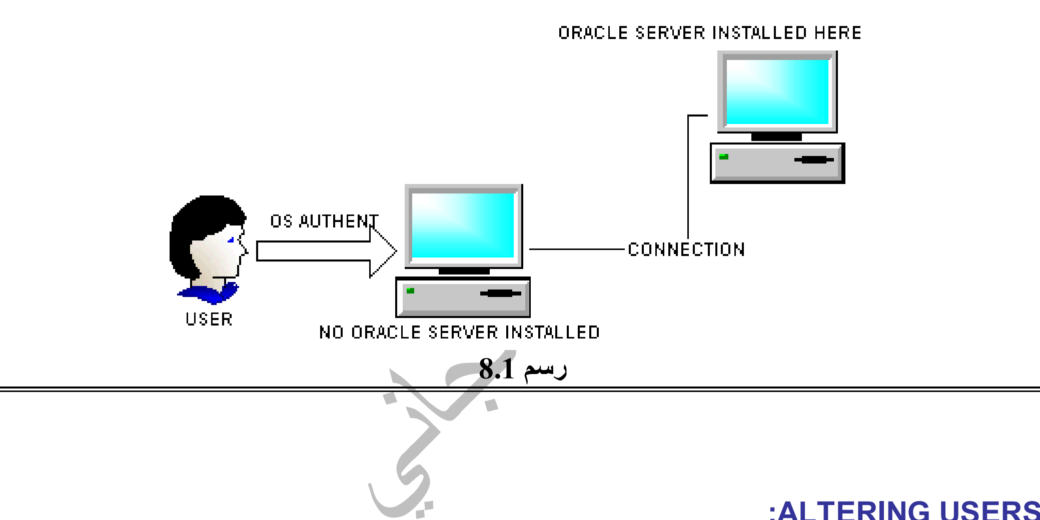
عندما يطلب المستخدم Ahmad الدخول الى ال Database ، فليس مطلوباً من ادخال كلمة سر أو اسم مستخدم اذ يقوم الأوراكل بمعرفة اسم المستخدم في نظام التشغيل (Ahmad) وربطه مع ال Account الخاص به (OPS$Ahmad).

ملاحظة: تغيير قيمة ال OS_AUTHENT_PREFIX قد يؤدي الى منع المستخدميين من الدخول الى ال Database (الذين يسمح لهم الدخول الى Database بدون كلمة سر).

يمكن استخدام عامل أخر هو REMOTE_OS_AUTHENT لتشغيل أوغلق امكانية الإتصال بال Remote Database عبر نظام التشغيل (لا ينصح باستخدامه لإمكانية حدوث اختراق لل Database).

مثال:

مثال على استخدام REMOTE_OS_AUTHENT ، دخول المستخدم (لديه إمكانية الدخول الى ال Database عبر حماية نظام التشغيل) الى نظام تشغيل (مثل ال Unix) متصل مع Oracle Server الموجود على جهاز أخر، في حالة وضع قيمة العامل True، فإن المستخدم يستطيع الإتصال بال Database، أما في حالة القيمة False (هي ال Default) فإن المستخدم لا يستطيع الإتصال بال Database ويستطيع فقط اذا كان ال Oracle Server موجود على ذات الجهاز الذي دخل إليه.



رسم 8.1

:ALTERING USERS

يمكن استخدام جملة ALTER USER لكي يتم تغيير ال Default Tablespace أو ال Temporary Tablespace أو ال Quota.

مثال تطبيقي 8.7:

لتغيير ال Quota للمستخدم:

```
ALTER USER Ahmad
QUOTA 0 ON users;
```

ملاحظة: في حالة تغيير قيمة ال Quota الى صفر وكان للمستخدم Table في ال Users Tablespace، فإن هذه ال Table لا يمكن أن يكبر حجمها أبداً (لا تستطيع حجز مزيد من ال Extents).

أو عبر ال Console:

- ادخل الى ال Console عبر Standalone.
- اضغط على اسم ال Database لكي تظهر نافذة تطلب من اسم المستخدم وكلمة السر.
- أدخل اسم المستخدم وكلمة السر مع التاكد من اختيار SYSDBA عوضاً عن Normal.
- اضغط على + ال Security Manager.
- اضغط على + بجانب المجلد Users، ثم اختر اسم المستخدم.
- غير البيانات في النافذة اليمنى.
- اضغط على Apply.

DROPPING USERS:

يمكن حذف مستخدم من ال Database باستخدام القاعدة:

```
DROP USER user_name [CASCADE]
```

في القاعدة:

- **CASCADE**: باضافتها الى الجملة يتم حذف كل العناصر "Objects" التي كونها المستخدم في ال Schema الخاصة به.

ملاحظة: لا يمكن حذف مستخدم متصل بال Database. يمكن استعمال KILL SESSION لإخراج المستخدم ثم حذفه من ال Database.

مثال تطبيقي 8.8:

لحذف مستخدم:

```
DROP USER Ahmad;
```

QUERYING INFORMATION:

- **DBA_USERS**: يمكن الحصول على معلومات حول اسماء المستخدميين و ال Default Tablespace وال Temporary Tablespace و اسم ال Profile وتوقيت تكوين ال Profile وحالة ال Account لجميع المستخدميين و غيرها من المعلومات.

- **DBA_TS_QUOTAS**: توفر معلومات حول ال Quotas المخصصة للمستخدميين من ال Tablespaces.

الفصل التاسع

إدارة البريفليج و أوديوتين و الرولز

Managing PRIVILEGES & AUDITING & ROLES

PRIVILEGES

باستخدام ال Privileges يمكن وضع حدود لكل مستخدم في ال Database، مثال على ذلك حدود على المهام التي يستطيع تنفيذها، وحدود على البيانات التي يستطيع تعديلها. يمكن إعطاء Privileges للمستخدميين بشكل مباشر أو أن يتم عبر ال Roles ، كما يمكن إعطاء Privileges عامة لجميع المستخدميين وذلك عبر إعطاء ال Privileges الى مجموعة تعرف باسم Public وهي مجموعة ينتمي إليها جميع المستخدميين في ال Database.

يوجد نوعيين من ال Privileges هما:

- OBJECT Privileges: وضع حدود وقيود على العناصر "Objects".
- SYSTEM Privileges: وضع حدود على المهام التي يستطيع المستخدم تنفيذها في ال Database بشكل عام.

SYSTEM PRIVILEGES:

يوجد عدد كبير من ال System Privileges في ال Database يتم تخصيصها للمستخدمين باستخدام جملة GRANT ويتم إعادتها (إلغائها) من المستخدمين باستخدام جملة REVOKE. يمكن توزيع ال System Privileges الى ثلاثة أقسام ، قسم متعلق بال Database بشكل عام ، والقسم الثاني متعلق بال Schema الخاصة بالمستخدم ، أما القسم الثالث فمتعلق بال Schemas الخاصة بالمستخدميين الأخرين. نسق ال Privileges في النوع الثالث والرابع متشابه من حيث الجملة ولكن يضاف كلمة ANY للنوع الثالث، مثال على ذلك CREATE TABLE (النوع الثاني) و CREATE ANY TABLE (النوع الثالث).

من ال System Privileges:

- ALTER SYSTEM
- ALTER SESSION
- ALTER DATABASE
- CREATE TABLESPACE
- CREATE TABLE
- CREATE PROCEDURE
- CREATE CLUSTER
- CREATE SESSION
- RESTRICTED SESSION
- CREATE ANY TABLE
- ALTER ANY TABLE
- DROP ANY TABLE
- SELECT ANY TABLE
- CREATE ANY INDEX
- ALTER ANY INDEX
- DROP ANY INDEX
- CREATE ANY SEQUENCE
- ALTER TABLESPACE
- DROP TABLESPACE
- UNLIMITED TABLESPACE

- SELECT ANY TABLE
- UPDATE ANY TABLE
- DELETE ANY TABLE
- BACKUP ANY TABLE
- INSERT ANY TABLE
- GRANT ANY PRIVILEGE
- GRANT ANY ROLE
- SYSOPER
- SYSDBA

ملاحظات:

1- لا يوجد CREATE INDEX privilege أو ANALYZE INDEX privilege ، لأنهما مشمولتين مع CREATE TABLE privilege ، بحيث اذا تم إعطاء CREATE TABLE للمستخدم يستطيع تكوين Indexes.

2- DROP TABLE Privilege من ضمن ال CREATE TABLE privilege.

3- DROP PROCEDURE privilege من ضمن ال CREATE PROCEDURE Privilege.

4- DROP CLUSTER privilege من ضمن ال CREATE CLUSTER privilege.

5- UNLIMITED TABLESPACE privilege توفر إمكانية استخدام أي مساحة في أي Tablespace ، ولكن تخصص فقط للمستخدميين ولا يمكن تخصيصها لل Roles.

6- لكي يستطيع المستخدم عمل Truncate لأي Table في أي Schema يجب أن يخصص له DROP ANY TABLE.

Protecting the Dictionary:

عندما يتم منح المستخدميين ALTER ANY TABLE أو SELECT ANY TABLE، يصبح بإمكان المستخدميين الدخول أو استخدام ال Dictionary Objects ، وهذا قد يشكل خطر على ال Database. يوفر الأوراكل العامل 07_DICTIONARY_ACCESSIBILITY من عوامل Initialization Parameter File الذي يمكن ال DBA في التحكم في ال Dictionary Objects، بحيث عند وضع قيمة العامل False ، يتم منع المستخدميين من الدخول الى جميع العناصر ضمن ال SYS Schema الذي يملك ال Dictionary.

:Granting System Privilege

يمكن إعطاء Privileges للمستخدم باستخدام القاعدة التالية:

```
GRANT system_privilege , system_privilege, …
TO {user_name  | role_name | PUBLIC} , { user_name | role_name | PUBLIC} , ..
[WITH ADMIN OPTION]
```

في القاعدة:

- **Role_name:** كما ذكرنا سابقاً أنه يمكن منح ال Privileges لل Roles.
- **WITH ADMIN OPTION:** يستطيع المستخدم الذي تلقى ال Privileges إعطاءها لمستخدميين أخريين.

مثال:

لفهم WITH ADMIN OPTION:

يوجد ثلاث مستخدميين الأول هو ال DBA والثاني هو Ahmad والثالث هو Omar. قام ال DBA بإعطاء CREATE ANY TABLE للمستخدم Ahmad مع استخدام WITH ADMIN OPTION، في هذه الحالة يمكن للمستخدم Ahmad إعطاء CREATE ANY TABLE للمستخدم Omar. أما في حال عدم ذكر جملة WITH ADMIN OPTION ، فلا يمكن للمستخدم Ahamd اعطاء ال Privilege للمستخدم Omar ، ويجب على Omar الحصول على ال Privilege من (المستخدم الذي لدية إمكانية منح ال Privileges) ال DBA.

مثال تطبيقي 9.1:

```
GRANT CREATE ANY TABLE TO Ahmad, Omar;

GRANT ALTER ANY TABLE , ALTER ANY INDEX
TO Ali WITH ADMIN OPTION;
```

أو عبر ال Console:

- ادخل الى ال Console عبر Standalone.
- اضغط على اسم ال Database لكي تظهر نافذة تطلب من اسم المستخدم وكلمة السر.
- أدخل اسم المستخدم وكلمة السر مع التاكد من اختيار SYSDBA عوضاً عن Normal.
- اضغط على + ال Security Manager.
- اضغط على + بجانب المجلد Users، ثم اختر اسم المستخدم.
- اختر القائمة System Privileges.
- اختر ال Privilege من القائمة العليا ثم اضغط على المثلث المتجه الى أسفل.
- ضع علامة صح في قسم Admin Option اذا كنت تريد ذلك.
- اضغط على Apply.

:Revoking System Privilege

يمكن سحب أو إلغاء ال Privileges الممنوحة للمستخدم باستخدام القاعدة التالية:

```
REVOKE system_privilege , system_privilege, …
FROM {user_name  | role_name | PUBLIC} , { user_name | role_name |
PUBLIC} , ..
```

ملاحظة: في بعض الحلات عند سحب Privilege من المستخدم يحدث خلل في باقي عناصر المستخدم. مثال على ذلك الخلل الذي قد يحدثه إلغاء CREATE TABLE من المستخدم الذي قام بتكوين Tables وViews، في هذه الحالة يحدث خلل لل Views لأن ال Views معتمدة على ال Tables.

مثال:

عند عمل REVOKE ل Privilege وتم استخدام WITH ADMIN OPTION مع ال GRANT.

قام ال DBA بإعطاء CREATE ANY TABLE للمستخدم Ahmad مع استخدام WITH ADMIN OPTION، وقام المستخدم Ahmad بإعطاء CREATE ANY TABLE للمستخدم Omar.
عند حدوث أمر REVOKE من قبل ال DBA لل CREATE ANY TABLE الممنوحة ل Ahmad ، لا يحدث عملية REVOKE لل CREATE ANY TABLE من عند المستخدم Omar، أي أن Omar مازال يملك ال CREATE ANY TABLE.

مثال تطبيقي 9.2:

```
REVOKE CREATE ANY TABLE FROM Ahmad, Omar;
```

أو عبر ال Console:

- **ادخل الى ال Console عبر Standalone.**
- **اضغط على اسم ال Database لكي تظهر نافذة تطلب من اسم المستخدم وكلمة السر.**
- **أدخل اسم المستخدم وكلمة السر مع التاكد من اختيار SYSDBA عوضاً عن Normal.**
- **اضغط على + ال Security Manager.**
- **اضغط على + بجانب المجلد Users، ثم اختر اسم المستخدم.**
- **اختر القائمة System Privileges.**
- **اختر ال Privilege من القائمة السفلى ثم اضغط على المثلث المتجه الى أعلى.**
- **اضغط على Apply.**

OBJECT PRIVILEGES:

توفر ال Object Privilege إمكانية القيام بمهمات محددة على ال Table, View, Sequence, Procedure, Function, Package.

من ال Object Privilege:

SELECT: تمكن المستخدم من استخراج أو قراءة (Query) بيانات من ال Table,View, Sequence.

UPDATE: تمكن المستخدم من تعديل البيانات في ال Table, View.

DELETE: تمكن المستخدم من حذف البيانات في ال Table, View.

INSERT: تمكن المستخدم من إضافة بيانات الى ال Table, View.

ALTER: تمكن المستخدم من تعديل تكوين ال Table, Sequence, Procedure.

INDEX: تمكن المستخدم من تكوين Index في ال Table, View.

REFERENCES: تمكن المستخدم من تكوين Foreign Key في ال Table.

EXECUTE: تمكن المستخدم من تشغيل برامج ال PL/SQL مثل ال Procedure, Function.

Granting Object Privilege:

يمكن إعطاء Privileges للمستخدم باستخدام القاعدة التالية:

```
GRANT object_privilege, object_privilege, … | ALL [PRIVILEGES]
ON [schema.] object_name
TO {user_name | role_name | PUBLIC} , ….
[WITH GRANT OPTION]
```

في القاعدة:

- **ALL:** لإعطاء كل ال Object Privileges
- **WITH GRANT OPTION:** يستطيع المستخدم الذي تلقى ال Privileges إعطاءها لمستخدميين أخريين.

ملاحظات:

1- لكي يستطيع مستخدم إعطاء Object Privileges الى مستخدم أخر، يجب أن يكون ال Object في ال Schema الخاصة به أو تم إعطاءه Object Privilege مع استخدام WITH GRANT OPTION.

2- لكل مستخدم Object Privileges كاملة على ال Objects الموجودة في ال Schema الخاصة به.

مثال تطبيقي 9.3:

```
GRANT UPDATE ON hr.employees
TO Ahmad
WITH GRANT OPTION
```

أو عبر ال Console:

- ادخل الى ال Console عبر Standalone.
- اضغط على اسم ال Database لكي تظهر نافذة تطلب من اسم المستخدم وكلمة السر.
- أدخل اسم المستخدم وكلمة السر مع التاكد من اختيار SYSDBA عوضاً عن Normal.
- اضغط على + ال Security Manager.
- اضغط على + بجانب المجلد Users، ثم اختر اسم المستخدم الذي تريد أن تعطيه ال Privilege مثل PM.
- اختر القائمة Object Privileges.
- اختر من القائمة العليا اليسرى اسم المستخدم (مثل HR) الذي يملك ال Object المراد أعطاء Privilege لل PM. ثم اختر نوعية ال Object (مثل Tables) ثم اسم ال Objects (مثل Employees)
- اختر نوعية ال Privilege من القائمة اليمنى (مثل Update).
- اضغط على السهم المتجه الى أسفل.
- اضغط على Apply.

Revoking Object Privilege:

يمكن سحب أو إلغاء ال Privileges الممنوحة للمستخدم باستخدام القاعدة التالية:

```
REVOKE object_privilege, object_privilege, … | ALL [PRIVILEGES]
ON [schema.] object_name
FROM {user_name | role_name | PUBLIC} , ….
[CASCADE CONSTRAINTS]
```

ملاحظة: لكي تتم عملية ال Revoke ، يجب أن يقوم بالعملية ذات المستخدم الذي قام بعملية Grant.

في القاعدة:

- CASCADE CONSTRAINTS: لحذف ال Foreign Keys التي تكونت باستخدام REFERENCES privilege.

مثال:

عند عمل REVOKE ل Privilege وتم استخدام WITH GRANT OPTION مع ال GRANT.

قام المستخدم Ali بإعطاء SELECT TABLE للمستخدم Ahmad مع استخدام WITH GRANT OPTION، وقام المستخدم Ahmad بإعطاء SELECT TABLE للمستخدم Omar.
عند حدوث أمر REVOKE من قبل ال Ali لل SELECT TABLE الممنوحة ل Ahmad ، يحدث عملية REVOKE لل SELECT TABLE من عند المستخدم Omar، أي أن Omar يفقد SELECT TABLE ايضاً.

مثال تطبيقي 9.4

```
REVOKE SELECT ON hr.employees
FROM Ahmad, Omar;
```

أو عبر ال Console:

- ادخل الى ال Console عبر Standalone.
- اضغط على اسم ال Database لكي تظهر نافذة تطلب من اسم المستخدم وكلمة السر.
- أدخل اسم المستخدم وكلمة السر مع التاكد من اختيار SYSDBA عوضاً عن Normal.
- اضغط على + ال Security Manager.
- اضغط على + بجانب المجلد Users، ثم اختر اسم المستخدم الذي تريد استرجاع ال Privileges منه.
- اختر القائمة Object Privileges.
- اختر ال Privilege من القائمة السفلى ثم اضغط على المثلث المتجه الى أعلى.
- اضغط على Apply.

:QUERYING INFORMATION

للحصول على معلومات حول ال System Privileges:

- DBA_SYS_PRIVS: توفر معلومات حول ال System Privileges الممنوحة الى Users, Roles.
- DBA_TAB_PRIVS: توفر معلومات حول ال Object Privileges الممنوحة والمستخدم الذي منح ال Privileges وغيرها.
- SESSION_PRIVS: توفر معلومات حول ال System Privileges الممنوحة للمستخدم الحالى (Session).

AUDITING THE DATABASE

عملية ال Auditing هي عملية مراقبة مهمات ال Database المختلفة بحيث يتم مراقبة أي عمليات مشتبه في أنها غير مسموح فيها كأن يقوم مستخدم غير مسموح له بالدخول الى ال Database بالدخول وحذف بيانات من ال Database. يتم جمع بيانات ومعلومات حول المراقبة في AUD$ التابعة ل SYS schema ، أو يمكن جمع هذه المعلومات في ملف خارجي في نظام التشغيل "Operating System". لتسجيل المعلومات الناتجة عن عملية ال Auditing ، يجب وضع قيمة العامل AUDIT_TRAIL بالقيمة True أو DB ، أو يمكن استخدام ملف خارجي بوضع قيمة العامل OS، أما القيمة ال Default للعامل هي NONE. يقوم الأوراكل بعمل Auditing في بعض الحالات وإن كانت قيمة العامل None، مثل عند تشغيل و إغلاق ال Instance أو لمراقبة ال Privileges الخاصة لل DBA (أو كل مستخدم يملك Privileges مشابه لل DBA).
لل DBA حرية تحديد اذا ما كان يريد ان تحدث عملية ال Auditing عند المهمة المراد مراقبتها (By Access) أو لكل ال Session بغض النظر عن عدد المهمات التي تحدث(By Session).

باستخدام عملية ال Audit يمكن مراقبة:

- جمل ال SQL
- ال Privileges.
- العناصر "Objects".

ملاحظة: يطلق على AUD$ اسم Audit Trail وهي من نوع ال Base Tables وهي الوحيدة التي يمكن لل DBA تعديلها كما جاء ذكرها سابقاً.

اذا امتلئ ملف ال Audit Trail بحيث لا يمكن إضافة معلومات المراقبة الى الملف ، تتأثر جميع جمل ال SQL المراقبة وعوضاً أن تظهر النتائج الطلوبة ، تظهر أخطاء "Error Messages". ولذلك يجب تفرغة ملف ال Audit Trail بشكل مستمر. للتحكم في حجم ال Audit Trail ، ينصح باستخدام ال Auditing عند الضرورة فقط ، واختيار نوع المراقبة بدقة (ال SQL أو العناصر أو ال Privileges) ، وعدم إعطاء ال AUDIT ANY privileges لمستخدميين كثيرين واستخدام By Session عوضاً عن By Access.

ملاحظة: لحذف محتويات ال Audit Trail ، تستخدم جمل ال DELETE أو ال TRUNCATE.

يمكن حماية ملف ال Audit Trail بكتابة التالي:

```
AUDIT DELETE ON SYS.AUD$ BY ACCESS;
```

يفضل نقل ال AUD$ من ال SYSTEM TABLESPACE الى Tablespace أخرى، مع مراعاة اعادة تعريف ال Indexes على كل من ال Columns التالية (sessionid , ses$tid).

تذكر: يجب اعادة تعريف ال Indexes عند نقل ال Tables.

مثال تطبيقي 9.5:

مثال على القيام بعملية Auditing لجمل ال SQL:

لمراقبة جمل ال DDL التي تنفذ على ال Tables :

```
AUDIT TABLE;
```

مثال على القيام بعملية Auditing لل Privileges:

```
AUDIT CREATE ANY SEQUENCE;
```

مثال على القيام بعملية Auditing لل Objects:

```
AUDIT UPDATE, DELETE ON HR.EMPLOYEES;
```

AUDITING OPTIONS:

يمكن استخدامها لتحديد عملية المراقبة بشكل أكبر.

- **WHENEVER SUCCESSFUL: في حالة مراقبة جمل ال SQL ، يتم جمع معلومات حول الجمل التي تمت بنجاح ولا يتم تسجيل معلومات حول الجمل التي لم تنجح في القيام بمهمتها.**
- **WHENEVER NOT SUCCESSFUL: في حالة مراقبة جمل ال SQL ، يتم جمع معلومات حول الجمل التي لم تنجح ولا يتم تسجيل معلومات حول الجمل التي نجحت في القيام بمهمتها.**
- **BY ACCESS.**
- **BY SESSION.**

مثال تطبيقي 9.6:

لمراقبة عمليات تكوين Sessions (دخول المستخدميين لل Database) :

```
AUDIT SESSION;
```

العمليات الناجحة فقط:

```
AUDIT SESSION WHENEVER SUCCESSFUL;
```

العمليات التي لم تنجح:

```
AUDIT SESSION WHENEVER NOT SUCCESSFUL;
```

NOAUDITING:

لايقاف عملية ال Aduiting ، تستخدم جملة NOAUDIT مع اضافة الجزء الذي تم مراقبته في عملية Auditing.

مثال تطبيقي 9.7:

للقيام بعملية Noauditing:

```
NOAUDIT UPDATE, DELETE ON HR.EMPLOYEES;
```

QUERYING INFORMATION:

للحصول على معلومات حول ال Auditing:

- **ALL_DEF_AUDIT_OPTS: توفر معلومات حول ال Deafult Audit Options.**
- **DBA_STMT_AUDIT_OPTS: توفر معلومات حول عمليات المراقبة على جمل ال SQL.**
- **DBA_PRIV_AUDIT_OPTS: توفر معلومات حول عمليات المراقبة على Privileges.**
- **DBA_OBJ_AUDIT_OPTS: توفر معلومات حول عمليات المراقبة على Objects.**

للحصول على المعلومات الناتجة من عملية ال Auditing:

- **DBA_AUDIT_TRAIL: جميع المعلومات الناتجة من عملية ال Auditing.**
- **DBA_AUDIT_OBJECT: المعلومات الناتجة من عملية ال Auditing على Privileges.**
- **DBA_AUDIT_STATMENT: المعلومات الناتجة من عملية ال Auditing على جمل ال SQL.**
- **DBA_AUDIT_SESSION: المعلومات الناتجة من عملية ال Auditing على دخول و خروج المستخدميين من ال Database.**

MANAGING ROLES

تسهل ال Roles عملية إدارة ال Privileges لأن ال Role عبارة عن مجموعة من ال Privileges تعطى أو تلغى من المستخدمين دفعة واحدة. مثال على ذلك اذا أراد ال DBA إعطاء عشرة Privileges لكل مستخدم جديد في ال Database ، فيجب على ال DBA كتابة جملة GRANT عشر مرات ، وكذلك الحال مع جملة REVOKE ، اما عند استخدام ال Role ، يتم كتابة جملة GRANT أو REVOKE مرة واحدة فقط. يمكن أن يتكون ال Role من ال System Privileges وال Object Privileges معاً ، ولا ينتمى ال Role الى أي Schema ، ويتم تخزين بيانات حول تكوينه في ال Data Dictionary.

:CREATING ROLES

يمكن تكوين Roles باستخدام القاعدة التالية:

```
CREATE ROLE role_name [NOT IDENTIFIED
 | IDENTIFIED { BY password | EXTERNALLY }
```

في القاعدة:

- NOT IDENTIFIED: لا يوجد حماية على ال Role.
- IDENTIFIED: يوجد حماية على ال Role مثل استخدام كلمة السر أو غيرها.
- EXTERNALLY: يستخدم حماية خارجية لل Role مثل حماية نظام التشغيل.

ملاحظة: يوجد أنواع حماية أخرى لل Role عدا كلمة السر أو Externally لا مجال للتطرق لها في الكتاب.

مثال تطبيقي 9.8:

لتكوين Role:

```
CREATE ROLE newusers;
```

أو:

```
CREATE ROLE newusers2 IDENTIFIED BY n6iut30o;
```

أو عبر ال Console:

- ادخل الى ال Console عبر Standalone.
- اضغط على اسم ال Database لكي تظهر نافذة تطلب من اسم المستخدم وكلمة السر.
- أدخل اسم المستخدم وكلمة السر مع التاكد من اختيار SYSDBA عوضاً عن Normal.
- اضغط على + ال Security Manager.
- اضغط على بالزر اليمين للفأرة على المجلد Roles ، ثم اختر من القائمة Create.
- ادخل اسم ونظام حماية ال Role ثم اضغط على Create.

ملاحظات:

1- بعد تكوين ال Role مباشرة ، تكون ال Role خالية تماماً من أي Privileges ويمكن استخدام جملة GRANT لإعطاء ال Role مجموعة من ال Privileges.

2- لتكوين Role ، يجب أن يكون للمستخدم CREATE ROLE privilege.

:PREDEFINED ROLES

عند تكوين ال Database ، يتم تكوين مجموعة من ال Roles بشكل أتوماتيكي منها:

- **CONNECT: ال Privileges ضمن هذا ال Role تتضمن امكانية الإتصال بال Database وتكوين Session ، امكانية تكوين كل من Tables, Views, Synonym, Cluster, Database Link.**

- **RESOURCE: ال Privileges ضمن هذا ال Role تتضمن امكانية تكوين كل من Tables Cluster, Sequence, Functions, Procedures, Triggers.**

- **SELECT_CATALOG_ROLE: ال Privileges ضمن هذا ال Role تتضمن امكانية استخراج بيانات ال Data Dictionary (عمل Query على ال Tables وال Views).**

- **DELETE_CATALOG_ROLE: ال Privilege ضمن هذا ال Role تتضمن امكانية حذف بيانات (استخدام DELETE) من ال Data Dictionary.**

- **EXECUTE_CATALOG_ROLE: ال Privilege ضمن هذا ال Role تتضمن امكانية استخدام جملة EXECUTE في ال Data Dictionary.**

- **EXP_FULL_DATABASE: ال Privilege ضمن هذا ال Role تتضمن امكانية عمل عملية Export لل Database.**

- **EXP_FULL_DATABASE: ال Privilege ضمن هذا ال Role تتضمن امكانية عمل عملية Import لل Database.**

ملاحظة: سوف يتم التطرق على عمليات ال Export و ال Import في دراسة الإمتحان الثالث من امتحانات ال Oracle9i.

تذكر: من أنواع ال Data Dictionary Views نوع _DBA الذي يمكن أن للمستخدم الدخول الى بياناته اذا كان يملك ال SELECT_CATALOG_ROLE.

:MODIFYING ROLES

يمكن تغيير نظام حماية ال Role أو حذف نظام الحماية باستخدام جملة ALTER ROLE كما في المثال 9.9.

ملاحظة: لتعديل نظام حماية ال Role ، يجب أن يكون للمستخدم ALTER ANY ROLE privilege.

مثال تطبيقي 9.9:

لتعديل نظام حماية ال Role:

ALTER ROLE newusers2 IDENTIFIED EXTERNALLY;

أو احذف نظام الحماية:

ALTER ROLE newusers NOTIDENTIFIED;

أو عبر ال Console:

- **ادخل الى ال Console عبر Standalone.**
- **اضغط على اسم ال Database لكي تظهر نافذة تطلب من اسم المستخدم وكلمة السر.**
- **أدخل اسم المستخدم وكلمة السر مع التاكد من اختيار SYSDBA عوضاً عن Normal.**
- **اضغط على + ال Security Manager.**
- **اضغط على + بجانب المجلد Roles ، ثم اختر اسم ال Role المراد تعديله.**
- **في النافذة اليمنى ، اختر نظام الحماية من قائمة Authentication.**
- **اضغط على Apply.**

:ASSIGNING PRIVILEGES TO ROLES

بعد تكوين ال Role مباشرة ، تكون ال Role خالية تماماً من أي Privileges ويمكن استخدام جملة GRANT التي استخدمت لإعطاء Privileges الى المستخدميين مع ذكر اسم ال Role بجانب كلمة TO عوضاً عن اسم المستخدم أو ال Public كما في المثال 9.10.

مثال تطبيقي 9.10:

لإعطاء Privileges الى ال Roles:

GRANT CREATE ROLE TO adminusers;

GRANT SELECT ON HR.EMPLOYEES TO newusers2;

أو عبر ال Console:

- ادخل الى ال Console عبر Standalone.
- اضغط على اسم ال Database لكي تظهر نافذة تطلب من اسم المستخدم وكلمة السر.
- أدخل اسم المستخدم وكلمة السر مع التاكد من اختيار SYSDBA عوضاً عن Normal.
- اضغط على + ال Security Manager.
- اضغط على + بجانب المجلد Roles ، ثم اختر اسم ال Role المراد إضافة Privileges له.
- أختر إما قائمة System Privileges أو قائمة Object Privileges ، ثم اتبع ذات الخطوات التي تم فيها أعطاء المستخدميين ال Privileges.
- اضغط على Apply.

ASSIGNING ROLES TO USERS:

يمكن إعطاء المستخدميين ال Roles بواسطة جملة GRANT التي استخدمت لإعطاء Privileges الى المستخدميين مع ذكر اسم ال Role بجانب كلمة GRANT عوضاً عن اسم ال Privilege كما في المثال 9.11.

مثال تطبيقي 9.11:

لإعطاء Privileges الى ال Roles:

```
GRANT adminusers TO Omar WITH ADMIN OPTION;

GRANT newusers TO Khaled;
```

كما يمكن دمج Roles ببعضها.

```
GRANT newusers TO newusers2;
```

ملاحظات:

1- المستخدم الذي يقوم بتكوين ال Role لديه الحق في WITH ADMIN OPTION على ذلك ال Role.

2- المستخدم الذي يملك GRANT ANY ROLE privilege يستطيع إعطاء أو إلغاء ال Roles من المستخدميين.

3- يمكن تحديد عدد ال Roles التي يمكن أن تكون في الحالة Enable باستخدام عامل من عوامل ال Intialazation Parameter File هو MAX_ENABLED_ROLES الذي قيمته ال Default هي 20.

:ASSIGNING DEFAULT ROLES

يمكن للمستخدم الواحد أن يملك أكثر من Role ، ولكن ليس بالضرورة أن تكون كلها في حالة **Enable**، ولكن يمكن جمع مجموعة من ال Roles لتمثل ال Default Role والذي يتم وضعه في حالة **Enable** بشكل أوتوماتيكي عند دخول المستخدم الى ال Database. لتعيين ال Deafult Role ، تستخدم القاعدة:

```
ALTER USER user_name DEFAULT ROLE
{ role_name , role_name | ALL [EXCEPT role_name , [role_name] ] | NONE }
```

في القاعدة:

- **ALL [EXCEPT…]**: يمكن تعيين كل ال Roles عدا المشموليين في جملة EXCEPT.

ملاحظة: يمكن استخدام هذه القاعدة فقط مع ال Roles الممنوحة مباشرة للمستخدم عبر استخدام GRANT ولا تنطبق على ال Roles المدمجة ضمن Roles.

مثال تطبيقي 9.12:

لتعيين Default Role للمستخدم:

```
ALTER USER Ahmad DEFAULT ROLE newusers, newusers2;

ALTER USER Khalid DEFAULT ROLE ALL;

ALTER USER Omar DEFAULT ROLE ALL EXCEPT adminusers;

ALTER USER Ali  DEFAULT ROLE NONE;
```


أو عبر ال Console:

- ادخل الى ال Console عبر Standalone.
- اضغط على اسم ال Database لكي تظهر نافذة تطلب من اسم المستخدم وكلمة السر.
- أدخل اسم المستخدم وكلمة السر مع التاكد من اختيار SYSDBA عوضاً عن Normal.
- اضغط على + ال Security Manager.
- اضغط على + بجانب المجلد Users ، ثم اختر اسم ال User المراد تعديله.
- في النافذة اليمنى ، اختر قائمة Role، ثم ضع علامة صح اسفل ال Deafult لل Role الذي تريده أن يكون Default.
- اضغط على Apply.

:ENABLING AND DISABLING ROLES

يمكن تحويل حالة ال Role بين Enable و Disable باستخدام جملة SET ROLE. لا يستطيع المستخدم استخدام ال Role عندما تكون في حالة Disable ، ويجب تحويل حالتها الى Enable لأستخدام ال Privileges داخلها. تؤثر ال SET ROLE فقط على ال Session ولا تكون التغييرات دائمة ، إنما تعود ال Role لوضعها الطبيعي بعد خروج المستخدم من ال Database. عندما يتطلب ال Role كلمة سر لتحويل حالته ، يحب كتابة كلمة السر في جملة SET ROLE ، ولا ينطبق الحال على ال Defualt Role وإن كان من ضمنه Roles تتطلب كلة سر لأن ال Deafult Role يتم تحويلها الى حالة Enable بشكل أوتوماتيكي.
يمكن تحويل حالة ال Role باستخدام القاعدة التالية:

```
SET ROLE
{ role_name [IDENTIFIED BY PASSWORD] [, role_name ….]
| ALL [EXCEPT [role_name, role_name,…]
|NONE }
```

مثال تطبيقي 9.13:

لتحويل حالة ال Role الى Enable:

```
SET ROLE newusers, departA_users;
```

أو

```
SET ROLE newusers2 IDENRIFIED BY n6iut30o;
```

لتحويل حالة جميع ال Roles الى Enable عدا ال adminusers:

```
SET ROLE ALL EXCEPT adminusers;
```

لتحويل حالة جميع ال Roles الى Disable:

```
SET ROLE NONE;
```

:REMOVING ROLES FROM USERS

يمكن إلغاء ال Roles من ال Users باستخدام القاعدة التالية:

```
REVOKE role_name , role_name, …
FROM { role_name | PUBLIC} , {role_name | PUBLIC} , ..;
```

مثال تطبيقي 9.14:

لإلغاء ال Role من ال Users:

```
REVOKE newusers FROM  Khaled;
```

أو عبر ال Console: ذات الطريقة المستخدمة في إلغاء Privileges من ال Users.

:DROPPING ROLES

يمكن حذف ال Roles من ال Database وبالتالي من ال Users باستخدام القاعدة التالية:

DROP ROLE role_name ;

ملاحظة: يحتاج المستخدم الى DROP ANY ROLE privilege لحذف أي Role في ال Database.

مثال تطبيقي 9.15:

لحذف ال Role من ال Database:

DROP ROLE newusers2;

أو باستخدام ال Console:

- ادخل الى ال Console عبر Standalone.
- اضغط على اسم ال Database لكي تظهر نافذة تطلب من اسم المستخدم وكلمة السر.
- أدخل اسم المستخدم وكلمة السر مع التاكد من اختيار SYSDBA عوضاً عن Normal.
- اضغط على + ال Security Manager.
- اضغط على + ال Roles واختر اسم ال Role المراد حذفه ثم بواسطة الزر اليمين للفأرة أختر من القائمة "Remove" (يمكن استخدام قائمة Object ثم Remove عوضاً عن الزر اليمين للفأرة).
- اختر Yes.

:QUERYING INFORMATION

للحصول على معلومات حول ال Roles:

- DBA_ROLES: توفر معلومات حول جميع ال Roles الموجودة في ال Database.
- DBA_ROLE_PRIVS: توفر معلومات حول ال Roles الممنوحة للمستخدمين أو المدمجة مع Roles أخرى.
- ROLE_ROLE_PRIVS: توفر معلومات حول ال Roles المدمجة مع Roles أخرى.
- DBA_SYS_PRIVS: توفر معلومات حول ال System Privileges الممنوحة للمستخدمين أو لل Roles.
- ROLE_SYS_PRIVS: توفر معلومات حول ال System Privileges الممنوحة لل Roles.
- ROLE_TAB_PRIVS: توفر معلومات حول ال Object Privileges الممنوحة لل Roles.
- SESSION_ROLES: توفر معلومات حول ال Enabled Roles للمستخدم الحالي.

الفصل العاشر

الدعم العالمي

GLOBALIZATION SUPPORT

GLOBALIZATION SUPPORT

المقصود بال Globalization Support ، إمكانية استخدام لغات عالمية وتقويمات عالمية في الأوراكل.

من المميزات التي يوفرها ال Globalization Support:

- **إمكانية حفظ واستخراج البيانات باللغة المحلية.**
- **إمكانية تشغيل ال Database Utilities باللغة المحلية.**
- **إمكانية عرض ال Error Messages باللغة المحلية.**
- **أمكانية استخدام بعض التقويمات المحلية مثل التقويم الياباني.**
- **إمكانية عرض الأرقام باللغة المحلية.**
- **إمكانية استخدام رموز العملات العالمية.**
- **إمكانية استخدام فارق التقويم الزمني بين الدول.**
- **إمكانية استخدام مجموعة من النظم لتعريف اللغات"Encoding Schemes" مثل Unicode.**
- **التحويل الأوتوماتيكي بين اللغات، أي استخدام المستخدم Client لغة مختلفة عن لغة السيرفر.**

ENCODING SCHEMES:

يوفر الأوراكل مجموعة من النظم لتعريف اللغات منها:

- **SINGLE-BYTE: يتم استخدام بايت واحد لتخزين حرف واحد. يوجد نوعان هما 7-bit (مثال US7ASCII) و 8-bit (مثال WE8DEC).**

- **VARYING-WIDTH: يمكن تخزين الحرف الواحد في أكثر من بايت وتستخدم هذه الطريقة غالباً في اللغات الأسيوية مثل الصينية واليابانية (مثال JEUC و AL32UTF8).**

- **FIXED-WIDTH: يتم تحديد عدد محدد من البايتات لكل حرف (مثال AL16UTF16).**

- **UNICODE: نظام معتمد في جميع أنحاء العالم لقدرته على تعريف جميع اللغات والرموز الخاصة والرموز المستخدمة في النشر (مثل © ®). يمكن أن تتكون ال Unicode من مجموعة من النظم المختلفة مثل Varying-Width (مثال UTF-8) وFixed-Width (مثال UTF-16).**

CHOOSING CHARACTER SET:

**خلال دراستنا لقاعدة CREATE DATABASE ، تعرفنا الى جملة CHARACTER SET التي تحدد نوعية النظام الخاص باللغة" Character Set" الذي سوف يستخدم في ال Database (مثل AL32UTF8). يستخدم نظام ال Character set لتخزين اسماء ال Tables وال Columns و Data في هيئة CHAR, VARCHAR2, LONG, CLOB. القيمة ال Default لل Character Set هي استخدام US7ASCII. لا يمكن تغيير ال Character Set بعد تكوين ال Database عدا في بعض الأستثنائات القليلة (مثلاً عندما يكون ال Character Set الجديد مطابق بشكل كبير للنوع القديم).
عملية انتقاء ال Character set المناسب تعتمد على الأحتياجات الحالية والمستقبلية معاً، أي في حالة وجود إحتمال استخدام لغات عديدة في المستقبل، يفضل استخدام Character Set قادر على استيعاب لغات عدة.**

أيضاً يجب الأخذ بالإعتبار نوعية ال Character Set المتوفرة في نظام التشغيل " Operating System" اذ أن الإختلاف بين النظاميين يؤدي الى عملية تحويل يقوم بهل الأوراكل مما قي يؤدي الى حدوث عبئ وضغط من جراء عملية التحويل ، مما قد يؤدي الى ضياع بيانات.
يعتبر استخدام Single-Byte أفضل من الناحية الفنية والأداء عن استخدام Varying-Width ولكن إمكانيات ال Single-Byte في تعدد اللغات محدودة جداً.

ملاحظة: لا يمكن استخدام Fixed-Width كنظام لل Character Set.

CHOOSING NATIONAL CHARACTER SET:

يوجد نوع إضافي في قاعدة CREATE DATABASE هو NATIONAL CHARACTER SET الذي يستخدم لتخزين البيانات في هيئة NCHAR, NVARCHAR2, NCLOB. القيمة ال Default لل National Character Set هي AF16UTF16. يمكن استخدام قيمتان لل National Character فقط هما قيمتا ال Unicode UTF-16 و UTF-8. لا يمكن تغيير ال National Character Set بعد تكوين ال Database عدا في بعض الأستثنـاءات القليلة.
عملية انتقاء ال National Character Set تعتمد على اللغة المراد استخدامها ، ففي حالة اللغات الأوروبية يتم حجز بين 1 Byte الى 2 Byte لكل حرف (أي أن الأحرف لا تحجز مساحة ثابتة بل متغيرة) ، ولذلك يفضل استخدام UTF-8 لأنها من نوع متغير Varying-Width (من 1 بايت الى بايت 3)، ولكن تعتبر ال UTF-16 أسرع وأقل عبئ في ال Database لأنها من النوع الثابت Fixed-Width.

USING NLS PARAMETERS:

يوفر الأوراكل مجموعة من العوامل يطلق عليها اسم National Language Support (NLS) تساعد على تحديد أمور مختلفة عالمياً مثل طريقة حفظ وعرض التاريخ والتوقيت، الرمز المستخدم للعملة المحلية، اليوم الأول في الأسبوع (السبت عند المسلمين والاثنين عند الغرب) وغيرها من الأمور.

يمكن تحديد عوامل ال NLS بثلاث طرق هي:

- عبر تعديل ال Initialization Parameter File.
- باستخدام ال Environment Variables (يستخدم في نظام الويندوز ال Registry)
- باستخدام جملة ال ALTER SESSION.

التعديل بواسطة جملة ALTER SESSION هو الأقوى ثم يليه استخدام ال Environment Variables. هذا يعني اذا تم تعديل أحد العوامل بواسطة ال Initialization Parameter File وبواسطة جملة ال ALTER SESSION ، فإن قيمة العامل تكون هي المحددة بجملة ال ALTER SESSION وليس القيمة المحددة بال Initialization Parameter File أو القيمة المحددة بال Environment Variables.

ملاحظة: بعض العوامل لا تتغير بواسطة ALTER SESSION أو باستخدام ال Environment Variables.

تذكر: جاء ذكر ال Environment Variables في الفصل الثالث.

:NLS Parameters

يوفر الأوراكل عدد من العوامل منها:

- **NLS_LANGUAGE: يستخدم في عرض اسماء الأيام والشهور، ويستخدم لعرض الرموز الخاصة بالتوقيت والتاريخ مثل قبل أو بعد الميلاد (AD, BC) أو التوقيت (PM, AM) ، ويستخدم كلغة الرسائل المعروضة في الأوراكل مثل رسائل الأخطاء "Error Messages" ، ويستخدم ايضاً في تحديد نوعية عملية ال Sorting.**
- **NLS_TERRITORY: يستخدم في تحديد طريقة عرض التاريخ ، واختيار رمز للعملة المحلية ، تحديد أول يوم في الأسبوع وتحديد قيم تعرف باسم قيم ال ISO.**
- **.NLS_LANG**

تشمل ال NLS_LANGUAGE قيم ال Default للعوامل التالية:

- **NLS_DATE_LANGUAGE: لتحديد اسماء الشهور والأيام والإختصارات المعمول بها (مثل اختصار يوم الإثنين بالغة الإنجليزية MON).**
- **NLS_SORT: لتغيير طريقة ترتيب البيانات "Sorting" في الأوراكل ، مثلاً اذا تم تحديد قيمتها بالتالي NLS_SORT =German فإن ترتيب البيانات سوف يكون طبقاً للأبجدية الألمانية.**

أما ال NLS_TERRITORY فتشمل قيم ال Default للعوامل التالية:

- **NLS_CURRENCY: لإختيار رمز للعملة المحلية.**
- **NLS_ISO_CURRENCY: لإختيار رمز مختصر لعملية البلد مثل USD للدولار الأمريكي.**
- **NLS_DATE_FORMAT: لتحديد طريقة عرض التاريخ، مثلاً الأيام ثم الشهور ثم السنين.**
- **NLS_NUMERIC_CHARACTERS: لتحديد طريقة عرض الأرقام التي تحوي على فواصل مثل الأرقام الغير صحيحة (مثل 222.44.5).**

المقصود بأن العامل NLS_TERRITORY يشمل القيم ال Default للعوامل الأخرى ، بذلك هو عند تحديد قيمة ال NLS_TERRITORY ، فليس من الضروري تحديد قيم العوامل التي يشملها. أيضاً في حالة تحديد العامل NLS_TERRITORY والعوامل التي يشملها ، يتم تغيير قيم العوامل التي يشملها، حيث أن تغيير قيمة العامل NLS_TERRITORY يتم فقط في ال Environment Variables أو عبر جملة ال ALTER SESSION مما يجعله النوع الأقوى من العوامل التي يشملها التي تتغير عبر جميع الطرق. مثال على ذلك اذا تم تحديد قيمة العامل NLS_CURRENCY بالقيمة £ (الباوند) في ال Initialization Parameter File ولكن ايضاً تم تحديد قيمة العملة المحلية في NLS_TERRITORY بالقيمة ¥ (الين الياباني) فإن رمز العملية المحلية في الأوراكل يكون الين الياباني.

:NLS_LANG Parameter

يمكن استخدام قيم مختلفة لكل مستخدم باستخدام ال NLS_LANG ويتم تحديدها فقط عبر استخدام ال Environment Variables. تتكون من ثلاثة اقسام هي Language و Territory و Character Set، معرفة بالقاعدة التالية:

NLS_LANG = Langugae_Territory.Charset

في القاعدة:

- Language: هي موازية للعامل NLS_LANGUAGE ويتم تعريف قيمتها في الأوراكل عند تحديدها ولا يتم الأخذ بقيمة NLS_LANGUAGE ، اذ يعتبر العامل NLS_LANG هو الأقوى في الأوراكل.
- Territory: هي موازية للعامل NLS_TERRITORY ويتم تعريف قيمتها في الأوراكل عند تحديدها ولا يتم الأخذ بقيمة NLS_TERRITORY ، اذ يعتبر العامل NLS_LANG هو الأقوى في الأوراكل.
- Charset: يحدد نوعية ال Character Set للمستخدم فقط.

NLS and Functions:

خلال دراستك للإمتحان الأول "Introduction to SQL" استخدمت ال NLS Parameters داخل عمليات متعددة مثل جملة TO_CHAR ، وللتذكير راجع المثال 10.1.

مثال تطبيقي 10.1:

```
SELECT TO_CHAR (hire_date, 'DD.Mon.YYYY',
'NLS_DATE_LANGUAGE = GERMAN')
FROM HR.EMPLOYEES
```


يمكن استخدام العوامل التالية مع TO_CHAR:

- NLS_DATE_LANGUAGE
- NLS_NUMERIC_CHARACTERS
- NLS_CURRENCY
- NLS_ISO_CURRENCY
- NLS_CALENDAR

يمكن استخدام العوامل التالية مع TO_DATE:

- NLS_DATE_LANGUAGE
- NLS_CALENDAR

يمكن استخدام العوامل التالية مع TO_NUMBER:

- NLS_NUMERIC_CHARACTERS
- NLS_CURRENCY
- NLS_ISO_CURRENCY

QUERYING INFORMATION:

للحصول على معلومات حول ال NLS:

- NLS_DATABASE_PARAMETER: توفر معلومات عن ال Character Set وال National Character Set المستخدم في ال Database.

```
SELECT PARAMETER, VALUE
FROM NLS_DATABASE_PARAMETER
WHERE PARAMETER LIKE '%CHARACTERSET%';
```

- NLS_INSTANCE_PARAMETER: توفر معلومات عن قيم عوامل ال NLS المحددة في ال Initialization Parameter File.
- NLS_SESSION_PARAMETER: توفر معلومات عن قيم عوامل ال NLS المحددة لل Session فقط.
- V$NLS_VALID_VALUES: توفر معلومات حول جميع القيم المتوفرة المسموح استخدامها لتحديد كل من ال Language, Territory, Character Set.